George Grätzer

Practical LaTeX

Springer

Practical LaTeX

George Grätzer

Practical LaTeX

George Grätzer
Toronto, ON, Canada

To my family
and especially the little ones,
Emma (9),
Kate (7),
Jay (2)

Short Contents

Contents

Introduction

To learn LaTeX, you have to read some really heavy books, such as my big book on LaTeX[1] (I will refer to it as MiL4), all 600 plus pages of it. These books are so big because they cover all of LaTeX, from everyday use to complex documents, such as books, and the esoteric.

This book is a practical introduction to LaTeX. It covers only what is most used in everyday documents. If you want to learn how to typeset fine-tuning, read MiL4. Chances are slim that you would need this in your work.

We start with a lightning fast introduction to LaTeX.

Chapter 1 is *Mission Impossible*, introducing LaTeX documents and presentations in about 30 pages. After reading this chapter, you should be able to type your own documents and make your own presentations.

The other chapters delve deeper into the topics started in the first. Chapter 2 deals with typing text and Chapter 3 with text environments, such as lists and theorems. Chapter 4 deals with typing formulas and Chapter 5 with displayed formulas. The structure of a LaTeX document is discussed in greater detail in Chapter 6. LaTeX is so efficient to use because we can customize it to our needs, as discussed in Chapter 7.

We further develop our skills in making presentations in Chapter 8 and drawing illustrations in Chapter 9.

The text and math symbol tables are collected in Appendices A and B. We provide you with a pdf file SymbolTables.pdf (see Section 1.1.2), so you can have these tables handy on your computer's desktop.

Finally, in Appendix C, we show you how to use LaTeX on an iPad.

We achieve such a slim book by focusing on the contemporary and the practical. We don't write about legacy commands (such as \bf, use \textbf), environments (such as eqnarray, use align), and document classes (such as article, use amsart). There is no discussion of how to write a complex document such as a book, the fonts you can use, and of the various tools we have for long documents.

---

[1]*More Math into LaTeX*, 4th edition. Springer-Verlag, New York, 2007. ISBN-13: 978-0-387-32289-6

These topics would deserve separate *Practical* books. For further reading, see the file `FurtherReading.pdf` in the `samples` folder; see Section 1.1.2.

You will judge this book by how well it serves you. I selected the topics based on my experience writing articles and books in LaTeX and about LaTeX, and running an international math journal. I believe that the topics you need to type average size LaTeX documents are covered. If you have any thoughts about what else should be included, please let me know.

Acknowledgement

I received valuable advice for this book from William Adams, Jacques Crémer, Michael Doob, Alan Litchfield, Raul Martinez, Craig Platt, and Herbert Schulz.

Barbara Beeton is always there when I need her.

George Gratzer

E-mail:
gratzer@me.com
Home page:
http://server.maths.umanitoba.ca/homepages/gratzer/

CHAPTER

1

Mission Impossible

It happens to most of us. We live a happy life without LaTeX and then, all of a sudden, we have to do something urgent that requires it.

If you are a student, maybe your professor turned to you and said "I need the solutions to these exercises typed up and distributed to the class by tomorrow" and the solutions are chock-full of formulas, difficult to do in Word.

Or you are a researcher whose documents have always been typed up by a secretary. You have to attend a conference and give a presentation. Your secretary is gone due to a budget cut . . .

In my case, it was a letter (this was before e-mail) from the American Mathematical Society, in which they informed me that my paper, written in Word, was accepted for publication. The AMS will publish the paper in nine months. However a LaTeX version would be published in three months!

> The mission, should you choose to accept it, is to get started really fast in LaTeX. Our goal is to produce in LaTeX the little article printed on the next page.

Relax, this chapter will not self-destruct in five seconds.

G. Grätzer, *Practical LaTeX*, DOI 10.1007/978-3-319-06425-3_1,
© Springer International Publishing Switzerland 2014

A TECHNICAL RESULT
FOR CONGRUENCES OF FINITE LATTICES

G. GRÄTZER

ABSTRACT. We present a technical result for congruences on finite lattices.

1. INTRODUCTION

In some recent research, G. Czédli and I, see [1] and [2], spent quite an effort in proving that some equivalence relations on a planar semimodular lattice are congruences. The number of cases we had to consider was dramatically cut by the following result.

Theorem 1. *Let L be a finite lattice. Let δ be an equivalence relation on L with intervals as equivalence classes. Then δ is a congruence relation iff the following condition and its dual hold:*

(C_+) *If x is covered by $y, z \in L$ and $x \equiv y \pmod{\delta}$, then $z \equiv y + z \pmod{\delta}$.*

2. THE PROOF

We prove the join-substitution property: if $x \leq y$ and $x \equiv y \pmod{\delta}$, then

$$(1) \qquad\qquad x + z \equiv y + z \pmod{\delta}.$$

Let $U = [x, y + z]$. We induct on length U, the length of U.

Let $I = [y_1, y + z]$ and $J = [z_1, y + z]$. Then length I and length $J <$ length U. Hence, the induction hypothesis applies to I and $\delta \rceil I$, and we obtain that $w \equiv y + w \pmod{\delta}$. By the transitivity of δ, we conclude that

$$(2) \qquad\qquad z_1 \equiv y + w \pmod{\delta}.$$

Therefore, applying the induction hypothesis to J and $\delta \rceil J$, we conclude (1).

REFERENCES

[1] G. Czédli, *Patch extensions and trajectory colorings of slim rectangular lattices.* Algebra Universalis **88** (2013), 255–280.
[2] G. Grätzer, *Congruences of fork extensions of lattices.* Acta Sci. Math. (Szeged), **57** (2014), 417–434.

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF MANITOBA, WINNIPEG, MB R3T 2N2, CANADA
E-mail address, G. Grätzer: `gratzer@me.com`
URL, G. Grätzer: `http://tinyurl.com/gratzerhomepage`

1.1 Getting started

*1.1.1 Your L*A*TEX*

Are you sitting in front of your computer, your LaTeX implementation up and running? If you use a UNIX computer, you surely are. If you are in front of a PC (with the Windows operating system) or a Mac, point your Internet browser at `tug.org`. Choose to download MikTeX for a PC and MacTeX for a Mac. Follow the easy instructions (and be patient, these are big downloads) and you are done.

Even better, find a friend who can help.

On a PC, `work\test` refers to the subfolder `test` of the folder `work`. On a UNIX computer and on a Mac, `work/test` designates this subfolder. To avoid having to write every subfolder twice, we use `work/test`, with apologies to our PC readers.

1.1.2 Sample files

We work with a few sample documents. Download them from `CTAN.org`, search for `Practical LaTeX`, or go to the Springer page for this book, and click on the link:
<http://extras.springer.com/2014/978-3-319-06424-6+>

I suggest you create a folder, `samples`, on your computer to store the downloaded sample files, and another folder called `work`, where you will keep your working files. Copy the documents from the `samples` to the `work` folder as needed. *In this book, the* `samples` *and* work *folders refer to the folders you have created.*

One of the sample files is `sample.sty`. Make sure it is in the `work` folder when you typeset a sample document.

1.1.3 Editing cycle

Watch a friend type a document in LaTeX and learn the basic steps.

1. *A text editor is used to create a L*A*TEX source file*. A source file might look like this:
   ```
   \documentclass{amsart}
   \begin{document}
   Then $\delta$ is a congruence relation. I can type formulas!
   \end{document}
   ```
   Note that the source file is different from a typical word processor file. All characters are displayed in the same font and size.

2. *Your friend "typesets" the source file* (*tells the application to produce a typeset version*) *and views the result on the monitor*:

   Then δ is a congruence relation. I can type formulas!

3. *The editing cycle continues.* Your friend goes back and forth between the source file and the typeset version, making changes and observing the results of these changes.

4. *The file is viewed/printed.* View the typeset version as a pdf file, print it if necessary, to create a paper version.

If LaTeX finds a mistake when typesetting the source file, it records this in the `log file`. The `log window` (some call it *console*) displays a shorter version.

Various LaTeX implementations have different names for the source file, the text editor, the typeset file, the typeset window, the `log` file, and the `log` window. Become familiar with these names, so you can follow along with our discussions.

1.1.4   Typing the source file

A source file is made up of *text, formulas*, and *instructions* (*commands*) *to* LaTeX.

For instance, consider the following variant of the first sentence of this paragraph:

```
A source file is made up of text, formulas (e.g.,
$\sqrt{5}$), and \emph{instructions to} \LaTeX.
```

This typesets as

A source file is made up of text, formulas (e.g., $\sqrt{5}$), and *instructions to* LaTeX.

In this sentence, the first part

```
A source file is made up of text, formulas (e.g.,
```

is text. Then

```
$\sqrt{5}$
```

is a formula

```
), and
```

is text again. Finally,

```
\emph{instructions to} \LaTeX.
```

are instructions. The instruction \emph is a *command with an argument,* while the instruction \LaTeX is a *command without an argument.* Commands, as a rule, start with a backslash (\ ) and tell LaTeX to do something special. In this case, the command \emph emphasizes its *argument* (the text between the braces). Another kind of instruction to LaTeX is called an *environment.* For instance, the commands

```
\begin{center}    \end{center}
```

enclose a `center` environment; the *contents* (the text typed between these two commands) are centered when typeset.

In practice, text, formulas, and instructions (commands) are mixed. For example,

```
My first integral: $\int \zeta^{2}(x) \, dx$.
```

is a mixture of all three; it typesets as

My first integral: $\int \zeta^2(x)\, dx$.

Creating a document in LaTeX requires that we type in the source file. So we start with the keyboard, proceed to type a short note, and learn some simple rules for typing text in LaTeX.

1.2   The keyboard

The following keys are used to type the source file:

```
a-z   A-Z   0-9
+ = * / ( ) [ ]
```

You can also use the following punctuation marks:

```
,   ;   .   ?   !   :   '   '   -
```

and the space bar, the Tab key, and the Return (or Enter) key.

Finally, there are thirteen special keys that are mostly used in LaTeX commands:

```
#   $   %   &   ~   _   ^   \   {   }   @   "   |
```

If you need to have these characters typeset in your document, there are commands to produce them. For instance, $ is typed as \$, the underscore, _, is typed as _, and % is typed as \%. Only @ requires no special command, type @ to print @; see Section A.3.3.

There are also commands to produce composite characters, such as accented characters, for example ä, which is typed as \"{a}. LaTeX prohibits the use of other keys on your keyboard unless you have special support for it. See the text accent table in Section A.2. If you want to use accented characters in your source file, then you must use the inputenc package.

Practical Tip 1. The text accent table looks formidable. Don't even dream of memorizing it. You will need very few. When you need a text accent, look it up. I know only one: \"a (LOL). If you have a name with accented characters, figure out once how to type it, and then any time you need it you can just copy and paste (chances are that the name is in your list of references).

1.3   Your first text notes

We start our discussion on how to type a note in LaTeX with a simple example. Suppose you want to use LaTeX to produce the following:

It is of some concern to me that the terminology used in multi-section math courses is not uniform.

In several sections of the course on matrix theory, the term "hamiltonian-reduced" is used. I, personally, would rather call these "hyper-simple". I invite others to comment on this problem.

To produce this typeset document, create a new file in your `work` folder with the name `textnote1.tex`. Type the following, including the spacing and linebreaks shown, but not the line numbers:

```
1    % Sample file: textnote1.tex
2    \documentclass{sample}
3
4    \begin{document}
5    It is of some concern to me    that
6    the terminology used in  multi-section
7     math courses is not uniform.
8
9    In several sections of the course on
10   matrix theory, the  term
11    ``hamiltonian-reduced'' is used.
12    I, personally, would rather call these
13   ``hyper-simple''. I invite others
14    to comment on this  problem.
15   \end{document}
```

Alternatively, copy the `textnote1.tex` file from the `samples` folder; see page .

The first line of `textnote1.tex` starts with %. Such lines are called *comments* and are ignored by LaTeX. Commenting is very useful. For example, if you want to add some notes to your source file and you do not want those notes to appear in the typeset version of your document, begin those lines with a %. You can also comment out part of a line:

```
simply put, we believe % actually, it's not so simple
```

Everything on the line after the % character is ignored by LaTeX.

Line 2 specifies the *document class*, `sample` (the special class we provided for the sample documents), which controls how the document is formatted.

The text of the note is typed within the `document` environment, that is, between \begin{document} and \end{document}.

Now typeset `textnote1.tex`. You should get the typeset document as shown. As you can see from this example, LaTeX is different from a word processor. It disregards the way you input and position the text, and follows only the formatting instructions given by the document class and the markup commands. LaTeX notices when you put a blank space in the text, but it ignores *how many blank spaces* have been typed. LaTeX does not distinguish between a blank space (hitting the space bar), a tab (hitting the Tab key), and a *single* carriage return (hitting Return once). However, hitting Return twice gives a blank line; *one or more* blank lines mark the end of a paragraph.

LaTeX, by default, fully justifies text by placing a flexible amount of space between words—the *interword space*—and a somewhat larger space between sentences—the *intersentence space*. If you have to force an interword space, you can use the \␣ command (in LaTeX books, we use the symbol ␣ to mean a blank space). The ~ (tilde) command also forces an interword space, but with a difference: it keeps the words on the same line. This command produces a *tie* or *nonbreakable space*.

Note that on lines 11 and 13, the left double quotes are typed as two left single quotes and the right double quotes are typed as two right single quotes, apostrophes.

We numbered the lines of the source file for easy reference. Sometimes you may want the same for the typeset file. This is really easy. Just add the two lines

```
\usepackage{lineno}
\linenumbers
```

after the `\documentclass` line and you get:

It is of some concern to me that the terminology used in multi-section math courses is not uniform.

In several sections of the course on matrix theory, the term "hamiltonian-reduced" is used. I, personally, would rather call these "hyper-simple". I invite others to comment on this problem.

Next, we produce the following note:

January 5, 2014

From the desk of George Grätzer

February 7–21 *please* use my temporary e-mail address:

George_Gratzer@yahoo.com

Type in the source file, without the line numbers. Save it in your work folder as `textnote2.tex` (`textnote2.tex` can be found in the `samples` folder):

```
1    % Sample file: textnote2.tex
2    \documentclass{sample}
3
4    \begin{document}
5    \begin{flushright}
6       \today
7    \end{flushright}
8    \textbf{From the desk of George Gr\"{a}tzer}
9
10   February 7--21 \emph{please} use my
11   temporary e-mail address:
12   \begin{center}
13      \texttt{George\_Gratzer@yahoo.com}
14   \end{center}
15   \end{document}
```

This note introduces several additional text features of LaTeX:

- The `\today` command (in line 6) to display the date on which the document is typeset, so you will see a date different from the date shown above in your own typeset document; see also Section 7.1.7.

- The environments to *right justify* (lines 5–7) and *center* (lines 12–14) text.

- The commands to change the text style, including the \emph command (line 10) to *emphasize* text, the \textbf command (line 8) for **bold** text (text bold font), and the \texttt command (line 13) to produce typewriter style text. These are *commands with arguments.*

- The form of the LaTeX commands. As we have noted already, almost all LaTeX *commands* start with a backslash (\ ) followed by the *command name*. For instance, \textbf is a command and textbf is the command name. The command name is terminated by the first *non-alphabetic character*, that is, by any character other than a–z or A–Z.

Practical Tip 2. textnote2.tex is a file name but textbf1 is not a command name. \textbf1 typesets as **1**. Let's look at this a bit more closely. \textbf is a valid command. If a command needs an argument and it is not followed by braces, then it takes the next character as its argument. So \textbf1 is the command \textbf with the argument 1, and it typesets as **1**.

- The multiple role of hyphens: Double hyphens are used for number ranges. For example, 7--21 (in line 10) typesets as 7–21. The punctuation mark – is called an *en dash*. Use triple hyphens for the *em dash* punctuation mark—such as the one in this sentence.

- Special rules for special characters (see Section 1.2), for *accented characters*, and for some *European characters.* For instance, the accented character ä is typed as \"{a}. (But I confess, I always type my name as Gr\"atzer without the braces.)

See Appendix A, where all the text symbols are organized into tables. Recall that we also have the SymbolTables.pdf in the samples folder.

Practical Tip 3. Keep SymbolTables.pdf handy on your computer!

1.4   Lines too wide

LaTeX reads the text in the source file one line at a time and when the end of a paragraph is reached, LaTeX typesets the entire paragraph. Occasionally, LaTeX gets into trouble when trying to split the paragraph into typeset lines.To illustrate this situation, modify textnote1.tex. In the second sentence, replace term by strange term. Now save this modified file in your work folder using the name textnote1bad.tex (or copy the file from the samples folder).

Typesetting textnote1bad.tex, you obtain the following:

It is of some concern to me that the terminology used in multi-section math courses is not uniform.

In several sections of the course on matrix theory, the strange term "hamiltonian-reduced" is used. I, personally, would rather call these "hyper-simple". I invite others to comment on this problem.

The first line of paragraph two is more than $1/4$ inch too wide. In the `log` window, LaTeX displays the following messages:

```
Overfull \hbox (15.38948pt
too wide) in paragraph at lines 9--15 []\OT1/cmr/m/n/10 In
sev-eral sec-tions of the course on ma-trix the-ory, the strange
term ''hamiltonian-
```

It informs you that the typeset version of this paragraph has a line that is 15.38948 points too wide. LaTeX uses *points* (pt) to measure distances; there are about 72 points in 1 inch. Then identifies the source of the problem: LaTeX did not properly hyphenate the word `hamiltonian-reduced` because it (automatically) hyphenates a hyphenated word *only at the hyphen*.

What to do, when a line is too long?

Practical Tip 4. Your first line of defense: reword the offending line. Write

```
    The strange term ''hamiltonian-reduced'' is used
  in several sections of the course on matrix theory.
```

and the problem goes away.

Practical Tip 5. Your second line of defense: insert one or more *optional hyphen commands* (\-), which tell LaTeX where it can hyphenate the word. Write:

```
hamil\-tonian-reduced
```

1.5 *A note with formulas*

In addition to the regular text keys and the 13 special keys discussed in Section 1.2, two more keys are used to type formulas: `<` and `>`. The formula $2 < |x| > y$ (typed as `$2 < |x| >y$`) uses both. Note that such a formula, called *inline*, is enclosed by a pair of `$` symbols.

We begin typesetting formulas with the following note:

In first-year calculus, we define intervals such as (u, v) and (u, ∞). Such an interval is a *neighborhood* of a if a is in the interval. Students should realize that ∞ is only a symbol, not a number. This is important since we soon introduce concepts such as $\lim_{x \to \infty} f(x)$.

When we introduce the derivative

$$\lim_{x \to a} \frac{f(x) - f(a)}{x - a},$$

we assume that the function is defined and continuous in a neighborhood of a.

To create the source file for this mixed text and math note, create a new document with your text editor. Name it `formulanote.tex`, place it in the `work` folder, and type the following, without the line numbers (or simply copy `formulanote.tex` from the `samples` folder):

```
1    % Sample file: formulanote.tex
2    \documentclass{sample}
3
4    \begin{document}
5    In first-year calculus, we   define intervals  such
6    as  $(u, v)$ and $(u, \infty)$. Such an interval
7    is a \emph{neighborhood} of  $a$
8    if  $a$ is in the interval. Students should
9    realize that  $\infty$ is only a
10   symbol, not a number. This is important since
11   we soon introduce concepts
12    such as $\lim_{x \to \infty} f(x)$.
13
14   When we introduce the derivative
15   \[
16      \lim_{x \to a} \frac{f(x) - f(a)}{x - a},
17   \]
18   we assume that the function is defined and
19   continuous in a neighborhood of  $a$.
20   \end{document}
```

This note introduces several basic concepts of formulas in LaTeX:

- There are two kinds of formulas and environments in `formulanote.tex`:

  - *Inline* formulas; they open and close with $ or open with \(and close with \).

  - *Displayed* formula environments; they open with \[and close with \]. (We will introduce many other displayed formula environments in Section 1.8 and Chapter 5.)

- LaTeX uses its own spacing rules within formula environments, and completely ignores the white spaces you type, with two exceptions:

  - Spaces that terminate commands.
    So in `$\infty a$` the space is not ignored; `$\inftya$` produces an error.

  - Spaces in the arguments of commands that temporarily revert to regular text. `\text` is such a command; see Section 1.7.

  The white space that you add when typing formulas is important only for the readability of the source file.

- A math symbol is invoked by a command. For example, the command for ∞ is `\infty` and the command for $\to$ is `\to`. The math symbols are organized into tables in Appendix B; see also `SymbolTables.pdf` in the `samples` folder.

- Some commands, such as \sqrt, need *arguments* enclosed by { and }. To typeset $\sqrt{5}$, type $\sqrt{5}$, where \sqrt is the command and 5 is the argument. Some commands need more than one argument. To get

$$\frac{3+x}{5}$$

type
\[\frac{3+x}{5} \]

where \frac is the command, 3+x and 5 are the arguments.

Practical Tip 6. Keep in mind that many spaces equal one space in text, whereas your spacing is ignored in formulas, unless the space terminates a command.

1.6   Errors in formulas

Even in such a simple note there are opportunities for errors. To help familiarize yourself with some of the most commonly seen LATEX errors in formulas, we deliberately introduce mistakes into formulanote.tex.

Experiment 1   In line 6 of formulanote.tex, delete the third $ symbol; save the file under the name formulanotebad1.tex in the work folder (or copy the file from the samples folder).

Typeset formulanotebad1.tex. LATEX generates the following error message:
```
! Missing $ inserted.
<inserted text>
                $
l.6 as $(u, v)$ and  (u, \infty
                                )$. Such an interval
```
LATEX reads (u, \infty) as text; but the \infty command instructs LATEX to typeset a math symbol, which can only be done in a formula. So LATEX offers to put a $ in front of \infty while typesetting the source file—it does not put the $ in the source file itself. LATEX attempts a cure, but in this example it comes too late, because the formula *should* start just before (u.

Experiment 2   In line 16 of formulanote.tex, delete the second } symbol and save it under the name formulanotebad2.tex in the work folder. This introduces an error: the closing brace of the subscript (see page 13) is missing. Now typeset the note. You get the error message
```
Missing } inserted.
<inserted text>
                }
l.12 such as $\lim_{x \to \infty f(x)$
```
LATEX reports that a closing brace (}) is missing, but it is not sure where the brace should be. LATEX noticed that a subscript started with {, but it reached the end of the formula before finding a closing brace }. To remedy this, you must look in the formula for an opening brace { that is not balanced, and insert the missing closing brace }. Make the necessary change and typeset again to view the difference.

Experiment 3   In `mathnote.tex`, delete the two $ signs in line 19, that is, replace `$a$` by `a`. Typeset the file. It typesets with no errors. Here is the last line of the typeset file you get:

we assume that the function is defined and continuous in a neighborhood of a.

instead of

we assume that the function is defined and continuous in a neighborhood of a.

This is probably the error most often made by beginners. There is no error message by LaTeX and the typeset version looks good. Notice the difference in the shape of the letter a in the two cases. You need sharp eyes to catch such an error.

Practical Tip 7.   After an error is corrected, LaTeX can refuse to typeset your document. If your document is `document.tex`, look in the same folder for the *auxiliary file* `document.aux` that was created by LaTeX. Delete `document.aux` and **typeset twice**.

1.7   The building blocks of a formula

A formula (inline or displayed) is built from components. We group them as follows:

Arithmetic	Ellipses	Operators
Binomial coefficients	Integrals	Roots
Congruences	Math accents	Text
Delimiters	Matrices	

In this section, I describe each of these groups, and provide examples illustrating their use. Read carefully only the groups you need!

Arithmetic   We type the arithmetic operations $a + b$, $a - b$, $-a$, a/b, and ab in the natural way: `$a + b$`, `$a - b$`, `$-a$`, `$a / b$`, and `$a b$` (the spaces are typed only for readability).

If you wish to use $\cdot$ or $\times$ for multiplication, as in $a \cdot b$ or $a \times b$, use `\cdot` or `\times`, respectively. The formulas $a \cdot b$ and $a \times b$ are typed as `$a \cdot b$` and `$a \times b$`.

Displayed fractions, such as

$$\frac{1 + 2x}{x + y + xy}$$

are typed with `\frac`:

`\[ \frac{1 + 2x}{x + y + xy} \]`

Subscripts and superscripts Subscripts are typed with _ (underscore) and superscripts with ^ (caret). Subscripts and superscripts should be enclosed in braces, that is, typed between { and }. To get a_1, type a_{1}. Omitting the braces in this example causes no harm, but to get a_{10}, you *must* type a_{10}. Indeed, a_10 is typeset as $a_1 0$.

There is one symbol, the prime ('), that is automatically superscripted in a formula. To get $f'(x)$, just type $f'(x)$. (On many keyboards, the key looks like this: `)

Binomial coefficients Binomial coefficients are typeset with the \binom command. \binom{a}{b + c} is here inline: $\binom{a}{b+c}$, whereas

$$\binom{a}{b+c}$$

is the displayed version.

Congruences The two most important forms are

$$a \equiv v \pmod{\theta} \quad \text{typed as} \quad \$a \equiv v \pmod{\theta}\$$$
$$a \equiv v \pod{\theta} \quad \text{typed as} \quad \$a \equiv v \pod{\theta}\$$$

Delimiters Parentheses and square brackets are examples of delimiters. They are used to delimit some subformulas, as in $[(a*b)+(c*d)]^{2}$, which typesets as $[(a*b)+(c*d)]^2$. LaTeX can be instructed to expand them vertically to enclose a formula such as

$$\left(\frac{1+x}{2+y^2} \right)^2$$

which is typed as

\[\left(\frac{1 + x}{2 + y^{2}} \right)^{2} \]

The \left(and \right) commands tell LaTeX to size the parentheses correctly, relative to the size of the formula inside the parentheses; sometimes the result is pleasing, sometimes not.

Ellipses In a formula, the ellipsis is printed either as *low* (or *on-the-line*) *dots*:

$F(x_1, \ldots, x_n)$ is typed as $F(x_{1}, \dots, x_{n})$

or as *centered dots*:

$x_1 + \cdots + x_n$ is typed as $x_{1} + \dots + x_{n}$

Use \cdots and \ldots if \dots does not work.

Integrals The command for an integral is \int. The lower limit is specified as a subscript and the upper limit is specified as a superscript. For example, the formula $\int_0^\pi \sin x\, dx = 2$ is typed as

```
$\int_{0}^{\pi} \sin x \, dx = 2$
```

where \, is a spacing command (without it, the formula looks bad).

Math accents The four most frequently used math accents are:

$\bar{a}$	typed as	`$\bar{a}$`	$\hat{a}$	typed as	`$\hat{a}$`
$\tilde{a}$	typed as	`$\tilde{a}$`	$\vec{a}$	typed as	`$\vec{a}$`

Matrices You type the matrix

$$\begin{matrix} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{matrix}$$

with the \matrix command

```
\[ \begin{matrix}
    a + b + c & uv    & x - y & 27\\
    a + b     & u + v & z     & 134
   \end{matrix} \]
```

The matrix environment separates adjacent matrix elements within a row with ampersands (&). Rows are *separated* by new line commands, \\.

Practical Tip 8. Do not end the last row with a new line command.

The matrix environment has to appear within a formula, as a rule, in a displayed formula. It can be used in the align environment discussed in Section 1.8.3.

The matrix environment does not provide delimiters. Several variants do, including pmatrix and vmatrix. For example,

$$\mathbf{A} = \begin{pmatrix} a+b+c & uv \\ a+b & u+v \end{pmatrix} \begin{vmatrix} 30 & 7 \\ 3 & 17 \end{vmatrix}$$

is typed as follows:

```
\[ \mathbf{A} =
   \begin{pmatrix}
     a + b + c & uv\\
     a + b & u + v
   \end{pmatrix}
   \begin{vmatrix}
     30 & 7\\
     3 & 17
   \end{vmatrix} \]
```

As you can see, pmatrix typesets as a matrix between a pair of \left(and \right) commands, while vmatrix typesets as a matrix between a pair of \left| and \right| commands. There is also bmatrix for square brackets.

Operators To typeset the sine function, $\sin x$, type `$\sin x$`. Note that `$sin x$` would be typeset as $sinx$ (how awful). LaTeX calls \sin an *operator.* Section B.7 lists a number of operators. Some are just like \sin. Others produce a more complex display, for example,

$$\lim_{x \to 0} f(x) = 0$$

is typed as

`\[ \lim_{x \to 0} f(x) = 0 \]`

Large operators The command for *sum* is \sum and for *product* is \prod. The following examples,

$$\sum_{i=1}^{n} x_i^2 \quad \prod_{i=1}^{n} x_i^2$$

are typed as

`\[\sum_{i=1}^{n} x_{i}^{2}\ \ \prod_{i=1}^{n}x_{i}^{2}\]`

Sum and product are examples of *large operators.* They are typeset larger in a displayed formula. They are listed in Section B.7.1.

Roots \sqrt produces a square root. `$\sqrt{a + 2b}$` typesets as $\sqrt{a + 2b}$. The n-th root, $\sqrt[n]{5}$, requires the use of an *optional argument,* which is specified in brackets: `$\sqrt[n]{5}$`.

Text You can include text in a formula with a \text command. For instance,

$$a = b, \quad \text{by assumption,}$$

is typed as

`\[ a = b, \text{\quad by assumption}, \]`

where \quad is a spacing command.

1.8   Displayed formulas

1.8.1   Equations

The equation environment creates a displayed formula and automatically generates an equation number. The equation

$$\int_0^\pi \sin x \, dx = 2 \tag{1}$$

is typed as

```
\begin{equation}\label{E:firstIntegral}
  \int_{0}^{\pi} \sin x \, dx = 2
\end{equation}
```

The equation number, which is automatically generated, depends on how many other numbered displayed formulas occur before the given equation. You can choose to have equations numbered within each section—(1.1), (1.2), ..., in Section 1; (2.1), (2.2), ..., in Section 2; and so on—by including, in the preamble (see Section 1.9), the command

```
\numberwithin{equation}{section}
```

You can choose to have the equation numbers on the right; see the reqno option of the amsart document class in Section 6.6.5.

The equation* environment is the same as the displayed formula opened with \[and closed with \] we discussed in Section 1.5. Sometimes you may want to use equation* for the ease of deleting the * if you wish.

1.8.2   Symbolic referencing

To reference this formula without having to remember a number—which can change when you edit your document—give the equation a symbolic label by using the \label command and refer to the equation in your document by using the symbolic label, the argument of the \label command. In this example, I have called the first equation firstIntegral, and used the convention that the label of an equation starts with E:, so that the complete \label command is \label{E:firstIntegral}.

The number of this formula is referenced with the \ref command. Its page is referenced using the \pageref command. For example, to get

see (1) on page 15.

type (see Section 1.3 for ˜)

```
see~(\ref{E:firstIntegral}) on page~\pageref{E:firstIntegral}.
```

The \eqref command provides the reference number in parentheses. So the last example could be typed

```
see~\eqref{E:firstIntegral} on page~\pageref{E:firstIntegral}.
```

The \eqref command is smart. Even if the equation number is referenced in emphasized or italicized text, the reference typesets upright (in roman type).

The main advantage of this cross-referencing system is that when you add, delete, or rearrange equations, LaTeX automatically renumbers the equations and adjusts the references that appear in your typeset document. For bibliographic references, LaTeX uses the \bibitem command to define a bibliographic item and the \cite command to cite it.

Practical Tip 9. For renumbering to work, you have to typeset **twice**.

Practical Tip 10. It is a good idea to check the LaTeX warnings periodically in the `log` file. If you forget to typeset the source file twice when necessary, LaTeX issues a warning.

What happens if you misspell a reference, e.g., typing `\ref{E:FirstIntegral}` instead of `\ref{E:firstIntegral}`? LaTeX typesets **??**. There are two warnings in the `log` file:

```
LaTeX Warning: Reference 'E:FirstIntegral' on page 39
        undefined on input line 475.
```

for the typeset page and the other one close to the end:

```
LaTeX Warning: There were undefined references.
```

If a `\cite` is misspelled, you get **[?]** and similar warnings.

Absolute referencing

Equations can also be *tagged* by attaching a name to the formula with the `\tag` command. The tag replaces the equation number.

For example,

$$(\mathrm{Int}) \qquad \int_0^\pi \sin x \, dx = 2$$

is typed as

```
\begin{equation}
  \int_{0}^{\pi} \sin x \, dx = 2 \tag{Int}
\end{equation}
```

Tags are *absolute.* This equation is *always* referred to as (Int). Equation numbers, on the other hand, are *relative,* they can change when the file is edited.

1.8.3 Aligned formulas

LaTeX has many ways to typeset multiline formulas. We discuss three constructs in this section: *simple alignment, annotated alignment,* and *cases.* For more constructs, see Chapter 5. For all of them, see MiL4.

For simple and annotated alignment we use the `align` environment. Each line in the `align` environment is a separate equation, which LaTeX automatically numbers.

Simple alignment

Simple alignment is used to align two or more formulas. To obtain the formulas

$$(2) \qquad r^2 = s^2 + t^2,$$
$$(3) \qquad 2u + 1 = v + w^\alpha.$$

type the following, using \\ as the *line separator* and & as the *alignment point*:

```
\begin{align}
  r^{2}  &= s^{2} + t^{2},                    \label{E:Pyth}\\
  2u + 1 &= v + w^{\alpha}.                    \label{E:alpha}
\end{align}
```

💡 **Practical Tip 11.** In this displayed formula, \\ is a *line separator*, not a new line command. Do not place a \\ to terminate the last line!

These formulas are numbered (2) and (3) because they are preceded by one numbered equation earlier in this section.

The `align` environment can also be used to break a long formula into two or more parts. Since numbering both lines in such a case would be undesirable, you can prevent the numbering of the second line by using the \notag command in the second part of the formula.

For example,

$$(4) \qquad h(x) = \int \left(\frac{f(x) + g(x)}{1 + f^2(x)} + \frac{1 + f(x)g(x)}{\sqrt{1 - \sin x}} \right) \, dx$$

$$= \int \frac{1 + f(x)}{1 + g(x)} \, dx - 2 \tan^{-1}(x - 2)$$

is typed as follows:

```
\begin{align}
  h(x) &= \int \left( \frac{f(x) + g(x)}{1+ f^{2}(x)}
           + \frac{1+ f(x)g(x)}{\sqrt{1 - \sin x}}
           \right) \, dx\label{E:longInt}\\
         &= \int \frac{1 + f(x)}{1 + g(x) } \, dx
           - 2 \tan^{-1}(x-2)\notag
\end{align}
```

The rules for simple alignment are easy to remember.

Practical Rule ■ Simple alignments

- Use the `align` environment.

- *Separate* the lines with \\.

- In each line, indicate the alignment point with &, one & per line. If the alignment point is adjacent to an $=$, $+$, and so on, place the & *before* to ensure proper spacing.

- Place a \notag command in each line that you do not wish numbered.

- If no line should be numbered, use the `align*` environment.

- Place a \label command in each numbered line you can want to reference with \ref, \eqref, or \pageref.

Annotated alignment

Annotated alignment allows you to align formulas and their annotations, that is, explanatory text, separately:

$$
\begin{aligned}
(5) \qquad x &= x \wedge (y \vee z) && \text{(by distributivity)} \\
&= (x \wedge y) \vee (x \wedge z) && \text{(by condition (M))} \\
&= y \vee z
\end{aligned}
$$

This is typed as

```
\begin{align}
   x &= x \wedge (y \vee z)
   &&\text{(by distributivity)}\label{E:DoAlign}\\
   &= (x \wedge y) \vee (x \wedge z)
   &&\text{(by condition (M))} \notag\\
   &= y \vee z \notag
\end{align}
```

Practical Rule ■ **Annotated alignment**

The rules for annotated alignment are similar to the rules of simple alignment. In each line, in addition to the alignment point marked by `&`, there is also a mark for the start of the annotation: `&&`.

1.8.4   Cases

The `cases` construct is a specialized matrix. It has to appear within a math environment such as the `equation` environment or the `align` environment. Here is a typical example:

$$
f(x) =
\begin{cases}
-x^2, & \text{if } x < 0; \\
\alpha + x, & \text{if } 0 \leq x \leq 1; \\
x^2, & \text{otherwise.}
\end{cases}
$$

It is typed as follows:

```
\[ f(x)=
   \begin{cases}
     -x^{2},       &\text{if $x < 0$;}\\
      \alpha + x,  &\text{if $0 \leq x \leq 1$;}\\
      x^{2},       &\text{otherwise.}
   \end{cases} \]
```

The rules for using the `cases` environment are the same as for matrices. Separate the lines with `\\` and indicate the annotation with `&`.

1.9   The anatomy of a document

To begin, we use the sample document `firstdocument.tex` (in the `samples` folder) to examine the anatomy of an document.

Every LaTeX document has two parts, the preamble and the body. The *preamble* of a document is everything from the first line of the source file down to the line

```
\begin{document}
```

The *body* is the contents of the `document` environment. For a schematic view of a document, see Figure 1.1.

The preamble contains instructions affecting the entire document. The *only* required command in the preamble is the `\documentclass` command. There are other commands (such as the `\usepackage` commands) that must be placed in the preamble if they are used, but these commands do not have to be present in every document.

Here is the preamble and top matter of `firstdocument`:

```
%First document, firstdocument.tex
\documentclass{amsart}
\usepackage{amssymb,latexsym}
```
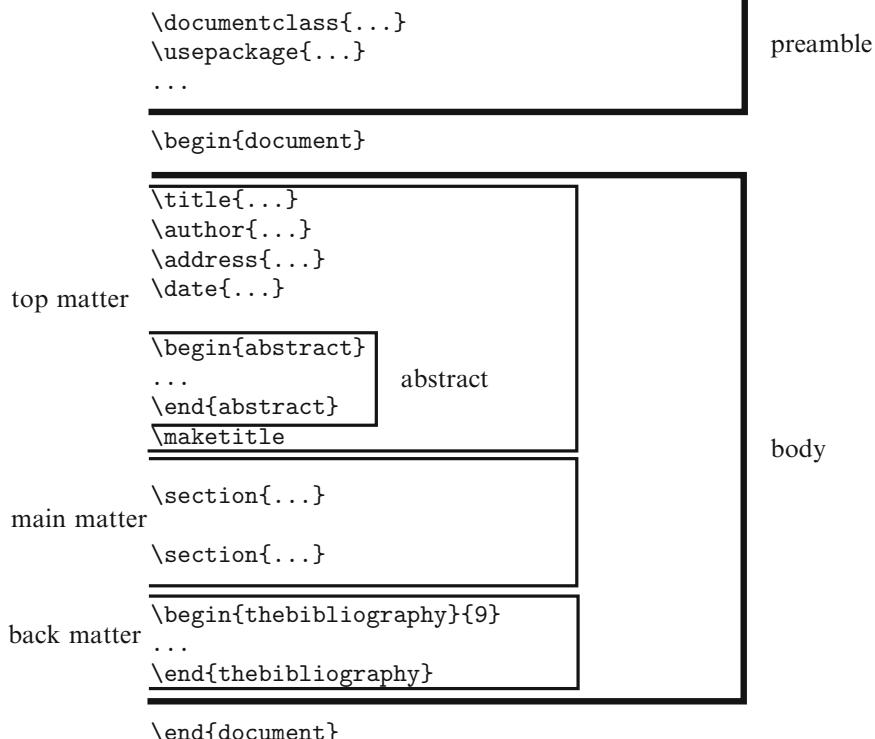
```
\documentclass{...}
\usepackage{...}
...                                                 preamble

\begin{document}

\title{...}
\author{...}
\address{...}
\date{...}
top matter
          \begin{abstract}
          ...                       abstract
          \end{abstract}
          \maketitle                                      body

          \section{...}
main matter
          \section{...}

          \begin{thebibliography}{9}
back matter ...
          \end{thebibliography}

\end{document}
```

Figure 1.1: A schematic view of a document.

```
\newtheorem{theorem}{Theorem}

\begin{document}
\title{A technical result\\ for congruences of finite lattices}
\author{G. Gr\"atzer}
\address{Department of Mathematics\\
  University of Manitoba\\
  Winnipeg, MB R3T 2N2\\
  Canada}
\email[G. Gr\"atzer]{gratzer@me.com}
\urladdr[G. Gr\"atzer]{http://tinyurl.com/gratzerhomepage}
\date{March 21, 2014}
\subjclass[2010]{Primary: 06B10.}
\keywords{finite lattice, congruence.}
\maketitle

\begin{abstract}
We present a technical result for congruences on finite lattices.
\end{abstract}
```

You find the source file in the samples folder and the typeset document on page 2.

To simplify the discussion in the book, we discuss only one document class: amsart. You may come across its predecessor article, which handles a limited set of commands for the preamble and the top matter and displays them differently. The article document class needs the help of the amsmath and amsthm packages (or equivalent) to handle proclamations (see Section 3.3) and many formulas, especially, displayed formulas (see Chapter 5). Some math journals require it. Use it if you have to.

1.10   LATEX error messages

Now that you are ready to type your first document, we give you some pointers on using LATEX.

You will probably make a number of mistakes in your first document. These mistakes fall into the following categories:

1. Typographical errors, which LATEX blindly typesets.

2. Errors in formulas or in the formatting of the text.

3. Errors in your instructions to LATEX, that is, in commands and environments.

Typographical errors can be corrected by viewing and spell checking the source file, finding the errors, and then editing the source file. Mistakes in the second and third categories may trigger errors during the typesetting process, such as the text errors of Section 1.4 and the formula errors in Section 1.6.

We now look at some examples of the third class of errors by deliberately introducing a number of mistakes into firstdocument.tex and examining the error messages.

Experiment 1.   In `firstdocument.tex`, go to line 19 by using your editor's `Go to Line` command and remove the closing brace so that it reads `\begin{abstract` When you typeset `firstdocument.tex`, LaTeX reports a problem:

```
{abstract We present a technical result for congruences on\ETC.
./firstdocument.tex:23:
Paragraph ended before \begin was complete.
<to be read again>
                    \par
l.23
```

Line 23 of the file is the line after `\maketitle`. The error message informs you that the environment name was not completed.

Runaway argument? is an error message that comes up often. It means that the argument of a command is either longer than expected or it contains material the argument cannot accept. Most often a closing brace solves the problem, as in this experiment.

Experiment 2.   Now correct line 19, then go to line 21 and change `\end{abstract}` to `\end{abstrac}` and typeset again. LaTeX informs you of another error:

```
./firstdocument.tex:21: LaTeX Error: \begin{abstract}
on input line 19 ended by \end{abstrac}.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...
l.21 \end{abstrac}
```

This is perfect. LaTeX correctly analyzes the problem and tells you where to make the change.

Experiment 3.   Correct the error in line 21, and introduce a new error in line 61. This line reads

```
z_1 \equiv y+ w \pmod{\delta}.
```

Change `\delta` to `\deta`. Now, when you typeset the document, LaTeX reports

```
./firstdocument.tex:61: Undefined control sequence.
<argument> {\operator@font mod}\mkern 6mu\deta

l.61 z_1 \equiv y+ w \pmod{\deta}
```

This mistake is easy to identify: `\deta` is a misspelling of `\delta`.

Experiment 4.   In line 38, delete the closing brace of the `\label` command. This results in a message:

```
Runaway definition?
->E:cover\text {If $x$ is covered by $y,z \in L$ and\ETC.
! File ended while scanning definition of \df@label.
<inserted text>
                }
<*> firstdocument.tex
```

Undo the change to line 38.

Experiment 5. Add a blank line following line 61:

```
x+ z = z + z_1 \equiv z + (y+ w) = y+ z \pmod{\delta},
```

This change results in the message

```
./firstdocument.tex:62: Missing $ inserted.
<inserted text>
                $
l.62
```

There can be no blank lines within a formula environment. LaTeX catches the mistake, but the message itself is misleading.

Experiment 6. Add a $ before \pmod in line 61 (such errors often occur when cutting and pasting formulas). You get the message:

```
./firstdocument.tex:61: Display math should end with $$.
<to be read again>
                   \penalty
l.61 z_1 \equiv y+ w $\pmod{\delta}
```

Maybe this could be more to the point?

Practical Tip 12. LaTeX's error messages are not very useful with displayed formulas. Comment out all but one of the lines to try to localize the problem.

Practical Tip 13. Typeset often.

Typesetting my book *First Steps into LaTeX* with the closing brace of the first \caption command on line 480 of the source file missing, I get the error message

```
! Text line contains an invalid character.
l.1227 ...pletely irreducible^^?
```

where the reference is to line 1227, about 700 lines removed from the actual error. However, if the only thing I did before typesetting was to insert that figure with its incorrect caption command, at least I would know where to look for errors. If you make a dozen changes and then typeset, you may not know where to start.

1.11 Adding an illustration

"And what is the use of a book," thought Alice, "without pictures or conversations?" I am not sure what to suggest about conversations, but illustrations we can tackle with ease.

Let us add an illustration, covers.pdf to firstdocument. First, add

```
\usepackage{graphicx}
```

as the fourth line of the document, to the preamble. This will enable LaTeX to tackle illustrations.

Secondly, add the following lines to `firstdocument.tex`, say, as the second paragraph of the introduction:

```
\begin{figure}[hbt]
{\centering\includegraphics{covers}}
\caption{Theorem~\ref{T:technical} illustrated}\label{F:Theorem}
\end{figure}
```

We place the illustration `covers.pdf` in the same folder as `firstdocument.tex`. That's it. You find `covers.pdf` and `firstdocumentill.tex` in the `samples` folder.

💡 **Practical Tip 14.** Make sure that the `\label` command follows the `\caption` command! You may have hard to explain troubles otherwise.

Most people in my field used the vector graphics application Adobe Illustrator to produce the PDF file for an illustration. In 2013 this became prohibitively expensive. Luckily, many reasonably priced alternatives are available. In Chapter 9, we discuss an alternative, Ti*k*Z, built for LaTeX.

1.12   Adding your own commands

Over time, LaTeX can be adjusted to fit your needs. You add packages to enable LaTeX to do new things (such as the `graphicx` package) and introduce your own commands to facilitate typing and make the source file more readable.

`firstdocumentmod.tex` (in the `samples` folder) adds two new commands to `firstdocument.tex`:

```
\newcommand{\pdelta}{\pmod{\delta}}
\DeclareMathOperator{\length}{length}
```

So instead of

```
$x \equiv y \pmod{\delta}$+
```

we can type

```
$x \equiv y \pdelta$
```

and instead of `length\,$U$`—how awful—we can type `$\length U$`. Notice how the spacing is now done by LaTeX!

We'll dedicate Chapter 5 to this topic. For more complete coverage, see Chapter 9 of MiL4.

1.13   Your errors: Davey's Dos and Don'ts

Based on his many years of experience correcting LaTeX articles for the journal *Algebra Universalis*, Brian Davey collected the LaTeX mistakes most often made by authors. Here are some items from his list, divided into three categories.

Commands

1. Place ALL user-defined commands and environments in the preamble!
   If you have trouble with user-defined commands, you then know where to find them.

2. Don't use \def; rather use \newcommand or \renewcommand.
   \def is the old TeX command. It is like \newcommand (see Section 1.12), but it can redefine an existing command. Redefining your own commands is bad enough, redefining a TeX command can be a disaster.

3. Do not simply type the name of an operator into a formula. Declare the appropriate operator; see Section 1.12.
   For instance, do not type $length I$; it typesets as $lengthI$. It should be $\length I$, typed as $\length I$. Of course, you have to add

   \DeclareMathOperator{\length}{length}

   to the preamble; see Section 1.9.

4. When you send a document to a coauthor or submit an article to a journal, remove all the user-defined commands not used.
   This is a real time saver for your coauthor and editor.

Text

1. Do not produce a list with horizontal and vertical spacing commands; see Section 2.7. Use a list environment; see Section 3.2.

2. Do not type numbers for citations and internal references. Use \cite{...} for citations and \ref{...} for references. For references to equations, use \eqref; see Section 1.8.2.

3. Do not number proclamations; see page 54. Use the standard amsart environments for theorems, and so on, and let LaTeX number them.

4. When writing a document for a journal requiring a document class file, **do not**

   (a) change any of the size parameters: for instance, do not use options like 12pt to change the font size or the \setlength command to change any parameter of the page size.

   (b) insert vertical white space via \bigskip, \smallskip, \vskip, \vspace, etc, nor via your own custom commands. Do not adjust horizontal space without a very good reason.

So if you want to display some text:

Please, display this text.

don't do this:

```
\medskip
\hspace*{6pt} Please, display this text.
\medskip
```

but rather

```
\begin{itemize}
\item[] Please, display this text.
\end{itemize}
```

or

```
\begin{quote}
Please, display this text.
\end{quote}
```

5. Do not leave a blank line before \end{proof}.

6. Do not use the geometry package.

Formulas

1. Don't use the symbol | in a set description, use the binary relation \mid; see Section 4.4.3.
   For instance, \{ x | x^2 < 2 \} typesets as $\{x|x^2 < 2\}$. The correct form is $\{x \mid x^2 < 2\}$, typed as $\{x \mid x^2 < 2\}$.

2. Don't put punctuation marks inside an inline math environment.
   For instance, $\sin x.$ typed as $\sin x.$; use $\sin x$. This typesets as $\sin x$. Notice the smaller space between "$\sin x$." and "typed" and the wider space between "$\sin x$." and "This".

3. Don't use two or more displayed formulas one after another. Use an appropriate environment such as \align, \alignat, \gather, and so on; see Section 5.1.

4. Don't use \left\{, \right\}, \left(, \right), and so on, by default (see page 13 for the commands \left and \right). Even when \left and \right do not change the size of the symbol, they add extra space after the closing delimiter.

5. Use `\colon` for functions. For instance, `$f(x) \colon x \to x^2$` which type-sets as $f(x)\colon x \to x^2$. If you type `$f(x) : x \to x^2$`, you get $f(x) : x \to x^2$.

6. Type a displayed formula environment (see Section 1.8) using `\[` and `\]` (or `equation*`) rather than the old TeX `$$` matched by `$$`. While display math produced via the latter does work properly most of the time, there are some LaTeX commands that do not; for example, `\qedhere`.

7. Do not use the `center` environment to display formulas.

8. Use `\dots` first and let LaTeX make the decision whether to use `\dots` or `\cdots`; see page 13. If LaTeX gets it wrong, then use `\cdots` or `\ldots`.

9. If you can, in inline formulas avoid constructs (for instance, $\overset{\text{up}}{\to}$) that disrupt the regular line spacing. Although LaTeX automatically leaves room for it, it does not look good, as a rule.

1.14   The anatomy of a presentation

Chances are, one of your first exposures to LaTeX was watching a *presentation*. The presenter used a pdf document produced by LaTeX and opened it with Adobe Reader. He went from "slide" to "slide" by pressing the space bar.

In LaTeX, you use a presentation package—really, a document class—to prepare such a PDF file. We use Till Tantau's BEAMER. Here is a part of the source file of our sample presentation, `firstpresentation.tex` (the complete presentation you can find at arxiv: 1309.6712).

```
\documentclass[leqno]{beamer}
\usetheme{Warsaw}
\DeclareMathOperator{\Princ}{Princ}
\begin{document}

\title[The order of principal congruences of a bounded lattice]
{The order of principal congruences\\ of a bounded lattice.\\
AMS Fall Southeastern Sectional Meeting\\
University of Louisville, Louisville, KY\\
October 5-6, 2013}
\author{G. Gr\"atzer}
\maketitle
```

`\usetheme{Warsaw}` provides a flavor. It is followed by the front page, providing the title and the author. Note that the `\title` has two parts. The first, in `[]`, is the short title, repeated on every slide. The second, in `{}`, is the title for the front page.

The rest of the presentation source file is divided into five *frames* with the structure:

```
\begin{frame}
\frametitle{}
\end{frame}
```

Each frame produces a "slide" (or more). Here is the third frame:

```
\begin{frame}
\frametitle{Theorem 1}
For a bounded lattice $L$, the order $\Princ K$ is bounded.
We now state the converse.
\pause
\begin{theorem}
Let $P$ be an order with zero and unit.
Then there is a bounded lattice~$K$ such that
\[ P \cong \Princ  K. \]
If $P$ is finite, we can construct $K$ as a finite lattice.
\end{theorem}
\end{frame}
\end{document}
```

The command \frametitle gives the slide its title. For instance, the third slide has the title: Theorem 1. In the body of the frame, you type regular LaTeX. There is only one new command to learn: \pause. Why the new command? I could achieve the same (so you might think) by using two frames:

```
\begin{frame}
\frametitle{Theorem 1}
For a bounded lattice $L$, the order $\textup{Princ}\,
K$ is bounded. We now state the converse.
\end{frame}

\begin{frame}
\frametitle{Theorem 1}
For a bounded lattice $L$, the order $\textup{Princ}\,
K$ is bounded. We now state the converse.

\begin{theorem}
Let $P$ be an order with zero and unit.
Then there is a bounded lattice~$K$ such that
\[ P \cong \textup{Princ}\, K. \]
If $P$ is finite, we can construct $K$ as a finite lattice.
\end{theorem}
\end{frame}
```

These two frames would produce the two slides of Figure 1.3. The contents seem fine, the placing is not! When you move from the first slide to the next, the text of the first slide jumps up, creating a jarring effect. Compare that to Figure 1.2. The second slide adds to the contents of the first without moving it.

Use \pause to display a list one item at a time. You can have more than one \pause in a frame.

Figure 1.2: Two slides with the \pause

Figure 1.3: Two slides without the \pause

2

Text

In Chapter 1, we briefly discussed how to type text in a document. Now we take up this topic more fully.

This chapter starts with Section 2.1, a discussion of words, sentences, and paragraphs. In Section 2.2, we are introduced to commands and environments.

A document can contain text symbols that cannot be found on your keyboard. In Section 2.3, we show how to get these symbols in our typeset documents by using commands.

Some other characters are defined by LaTeX as command characters. For example, the % character plays a special role in the source document. In Section 2.4.1, you will see how % is used to comment out lines. In Section 2.4.2, we introduce the command for footnotes.

In Section 2.5, we discuss the commands (and environments) for changing font shapes and sizes. In Section 2.6, you learn about lines, paragraphs, and pages. The judicious use of horizontal and vertical spacing is an important part of document formatting, and also the topic of Section 2.7. In Section 2.8, you learn how to typeset text in a "box", which behaves as if it were a single large character.

2.1   Words, sentences, and paragraphs

Text consists of words, sentences, and paragraphs. In text, *words* are separated by one or more spaces, which can include a single end-of-line character (see the rule, **Spacing in text**), or by parentheses and punctuation marks. A group of words terminated by a period, exclamation point, or question mark forms a *sentence*. A group of sentences terminated by one or more blank lines constitutes a *paragraph*.

2.1.1   Spacing rules

Here are the most important LaTeX rules about spaces in text in the source file.

Practical Rule ■   Spacing in text

1. Two or more spaces in text are the same as one.

2. A tab or end-of-line character is the same as a space.

3. A blank line, that is, two end-of-line characters separated only by spaces and tabs, indicates the end of a paragraph. The `\par` command does the same.

4. Spaces at the beginning of a line are ignored.

Rules 1 and 2 make cutting and pasting text less error-prone. However,

```
the number of    spaces separating words,
as long
```

and

```
the number of    spaces separating words
, as long
```

produce different results:

the number of spaces separating words, as long
the number of spaces separating words , as long

Notice the space between "words" and the comma in the second line. That space was produced by the end-of-line character in accordance with Rule 2.

Practical Tip 15. It is very important to maintain the readability of your source file. LaTeX does not care about the number of spaces or line length, but you, your coauthor, or your editor might.

2.1.2   Periods

LaTeX places a certain size space between words—the *interword space*—and a somewhat larger space between sentences—the *intersentence space.* To know which space to use, LaTeX must decide whether or not a period indicates the end of a sentence.

Practical Rule 1 ■ Period

To LaTeX, a period after a capital letter, for instance, A. or caT., signifies an abbreviation or an initial. Generally, every other period signifies the end of a sentence.

This rule works most of the time. When it fails—for instance, twice with e.g.—you need to specify the type of space you want, using the following two rules.

Practical Rule 2 ■ Period

If an abbreviation does not end with a capital letter, for instance, `etc.`, and it is not the last word in the sentence, then follow the period by an interword space (\␣) or a tie (˜), if appropriate.

Recall that \␣ provides an interword space.

```
The result was first published, in a first approximation,
in the Combin.\ Journal. The result was first published,
in a first approximation, in the Combin. Journal.
```

prints as

The result was first published, in a first approximation, in the Combin. Journal. The result was first published, in a first approximation, in the Combin. Journal.

Notice that Combin. in the first line is followed by a regular interword space. The incorrect intersentence space following Combin. in the second line is a little wider.

💡 **Practical Tip 16.** The `thebibliography` environment handles periods properly. You do not have to mark periods for abbreviations (in the form `.␣`) in the name of a journal, so

```
Acta Math. Acad. Sci. Hungar.
```

is correct.

The next rule contradicts Rules 1 and 2; consider it an exception.

Practical Rule 3 ■ Period

If a capital letter is followed by a period and is at the end of a sentence, precede the period with `\@`.

For example,

```
(1) follows from condition~H\@. We can proceed\\
(1) follows from condition~H. We can proceed
```

prints:

> (1) follows from condition H. We can proceed
> (1) follows from condition H. We can proceed

Notice that there is not enough space after H. in the second line.

Most typographers agree on the following rule:

Practical Rule 4 ■ Period

Add no space or a thin space (\,) within strings of initials and be consistent.

So W.H. Lampstone with no space or W. H. Lampstone with thin space is preferred over W. H. Lampstone.

2.2   *Commanding LaTeX*

How do you tell LaTeX to do something special for you, such as starting a new line, changing emphasis, or displaying the next theorem? You use *commands* and special pairs of commands called *environments*, both briefly introduced in Section 1.12.

Many commands have *arguments,* which are usually fairly brief. Environments have *contents*, the text between the \begin and \end commands. The contents of an environment can be several paragraphs long.

2.2.1   *Commands and environments*

The \emph{Careful!} *command* instructs LaTeX to emphasize its argument: *Careful!* The \& command has no argument. It instructs LaTeX to typeset &; see Section 1.2.

The center *environment* instructs LaTeX to center the contents, the text between the two commands \begin{center} and \end{center}. The body of the document (see Section 1.9) is the contents of the document environment and the abstract is the contents of the abstract environment.

Practical Rule ■ Environments

An environment starts with the command \begin{*name*} and ends with \end{*name*}. Between these two lines is the *contents* of the environment, affected by the definition of the environment.

Practical Rule ■ Commands

A LATEX command starts with a backslash, \, and is followed by the *command name.* The *name* of a command is either a *single non-alphabetic character* other than a tab or end-of-line character or a *string of letters,* that is, one or more letters.

So # and ' are valid command names. The corresponding commands \# and \' are discussed in Sections 1.2. More valid command names: input and date. However, input3, in#ut, and in␣ut are not valid names because 3, #, and ␣ should not occur in a multicharacter command name. Note that ␣ is a command name; the command \␣ produces a blank.

   LATEX has a few commands, for instance, $ (see Section 1.5) that do not follow this naming scheme, that is, they are not of the form \name.

Practical Rule ■ Command termination

LATEX finds the end of a command name as follows:

- If the first character of the name is not a letter, the name is the first character.

- If the first character of the name is a letter, the command name is terminated by the first nonletter.

If the command name is a string of letters, and is terminated by a space, then LATEX discards all spaces following the command name.

   While emph3 is an invalid name, \emph3 is not an incorrect command. It is the \emph command followed by the character 3, which is either part of the text following the command or the argument of the command.

   LATEX also allows some command names to be modified with ∗. Such commands are referred to as ∗-*ed commands.* Many commands have ∗-ed variants. \hspace* is an often-used ∗-ed command; see Section 2.7.1.

Practical Rule ■ Command and environment names

Command and environment names are *case sensitive.* \ShowLabels is not the same as \showlabels.

Practical Rule ■ Arguments

Arguments are enclosed in braces, { }.
Optional arguments are enclosed in brackets, [ ].

   Commands can have *arguments,* typed in braces immediately after the command. The argument(s) are used in processing the command. Accents provide very simple

examples. For instance, \'{o}—which produces ó—consists of the command \' and the argument o; see Sections 1.3 and 2.3.1. In \emph{together}, the command is \emph and the argument is together.

Some environments also have arguments. For example, the alignat environment (see Section 5.3.1) is delimited by the commands

\begin{alignat}{2} and \end{alignat}

The argument, 2, is the number of columns—it could be any number 1, 2, ... A command or environment can have more than one argument. The \frac command has two; $\frac{1}{2}$ typesets as $\frac{1}{2}$.

Some commands and environments have one or more *optional arguments,* that is, arguments that may or may not be present. The \sqrt command (see Section 1.7) has an optional argument for specifying roots other than the square root. To get $\sqrt[3]{25}$, type \sqrt[3]{25}. The \documentclass command has an argument, the name of a document class, and an optional argument, a list of options (see Section 6.2), for instance,

\documentclass[12pt,draft]{amsart}

Practical Tip 17. If you get an error when using a command, check that:

1. The command is spelled correctly, including the use of uppercase and lowercase letters.

2. You have specified all required arguments in braces.

3. Any optional argument is in brackets, not braces or parentheses.

4. The command is properly terminated.

5. The package providing the command is loaded with the \usepackage command.

Most errors in the use of commands are caused by breaking the termination rule. We illustrate some of these errors with the \today command, which produces today's date. You have already seen this command in Section 1.3. The correct usage is \today\␣ or \today{}. In the first case, \today was terminated by \␣, the command that produces an interword space. In the second case, it was terminated by the *empty group* { }.

If there is no space after the \today command, as in

\todayis␣the␣day

you get the error message

```
! Undefined control sequence.
l.6 \todayis
             the day
```

LaTeX thinks that \todayis is the command, and, of course, does not recognize it.

If you type one or more spaces after \today:

`\today␣␣is␣the␣day`

LATEX interprets the two spaces as a single space by the first space rule (see page 32), and uses that one space to delimit \today from the text that follows it. So LATEX produces

March 19, 2014is the day

💡 **Practical Tip 18.** If a command—or environment—can have an optional argument and none is given, and the text following the command starts with [, then type this as {[}.

This can happen, for instance, with the command \item (see page 54).

2.2.2   *Scope*

A command issued inside a pair of braces { } has no effect beyond the right brace. You can have many braces:

`{ ... { ... { ... } ... } ... }`

The innermost pair containing a command is the *scope* of that command. The command has no effect outside its scope. We can illustrate this concept using the \bfseries command that switches the font to boldface:

`{some text \bfseries bold text} no more bold`

typesets as

some text **bold text** no more bold

The commands \begin{*name*} and \end{*name*} bracketing an environment act also as a pair of braces and so delimit the scope. Also, $, \[, and \] are special braces.

Practical Rule ■ Braces

1. Braces must be balanced: An opening brace (left brace) must have a matching closing brace (right brace), and a closing brace (right brace) must have a matching opening brace.

2. Pairs of braces cannot overlap.

Violating the first brace rule generates warnings and error messages. If there is one more opening brace than closing brace, the document typesets, but you get a warning:

`(\end occurred inside a group at level 1)`

For two or more unmatched opening braces, you are warned that \end occurred inside a group at level 2, and so on.

💡 **Practical Tip 19.** Do not disregard such warnings even if the document is already correctly typeset. At a later time, such errors can have strange consequences.

2.2.3   *Types of commands*

It can be useful at this point to note that commands can be of various types.

Some commands have arguments, and some do not. Some commands effect change only in their arguments, while some commands declare a change. For instance, \textbf{This is bold} typesets the phrase This is bold in bold type: **This is bold** and has no effect on the text following the argument of the command. On the other hand, the command \bfseries declares that the text that follows should be bold. This command has no argument. I call a command that declares change a *command declaration.* So \bfseries is a command declaration, while \textbf is not. As a rule, command declarations are commands without arguments.

Commands with arguments are called *long*—or commands with long arguments— if their argument(s) can contain a blank line or a \par command; otherwise they are *short*—or commands with short arguments. For example, \textbf is a short command. So are all the top matter commands discussed in Section 6.6.1.

Fragile commands

As a rule, LaTeX reads a paragraph of the source file, typesets it, and then goes on to the next paragraph. Some information from the source file, however, is separately stored for later use.

Examples: the title of a document, which is reused as a running head (Section 6.6.1), table of contents (Sections 6.5.4), or footnote (Section 2.4.2).

These are *movable arguments,* and certain commands embedded in them must be protected from damage while being moved. LaTeX commands that need such protection are called *fragile.* The inline math delimiter commands \(and \) are fragile, while $ is not. In a movable argument, fragile commands must be protected with a \protect command. Thus \(f(x^{2}) \) is not appropriate in the title for a document, but

\protect \(f(x^{2}) \protect \)

is. To be on the safe side, you should protect every command that might cause problems in a movable argument. A user-defined command, declared with
\DeclareRobustCommand
is not fragile; it needs no protection. This command is like the \newcommand (see Section 7.1) but defines a *robust* command.

2.3   *Symbols not on the keyboard*

A typeset document can contain symbols that cannot be typed. Some of these symbols can even be available on the keyboard but you are prohibited from using them. In this section, we discuss the commands that typeset some of these symbols in text.

Quotation marks To produce single and double quotes, as in

'subdirectly irreducible' and "subdirectly irreducible"

type

```
'subdirectly irreducible' and ''subdirectly irreducible''
```

Here, ' is the left single quote and ' is the right single quote.

💡 **Practical Tip 20.** The double quote is obtained by typing the single quote key twice, and *not* by using the double quote key.

Dashes: hyphens A *hyphen*, -, is used to connect words:

Mean-Value Theorem

This phrase is typed with a single dash:

```
Mean-Value Theorem
```

Dashes: en dashes An *en dash*, –, is typed as `--` and is used for number ranges; for instance, the phrase see pages 23–45, is typed as

```
see pages~23--45
```

where ~ is a nonbreakable space (see Section 1.3), which is used to avoid having pages at the end of one line and 23–45 at the beginning of the next line.

Dashes: em dashes A long dash—called an *em dash*—is used to mark a change in thought or to add emphasis to a parenthetical clause, as in this sentence. The two em dashes in the last sentence are typed as follows:

```
A long dash---called an \emph{em dash}---is used
```

Note that there is no space before or after an en dash or em dash and en dash or em dash in a formula except in the argument of a \text command.

Ties or nonbreakable spaces A *tie* or *nonbreakable space* is an interword space that cannot be broken across lines. For instance, when referencing P. Neukomm in a document, you do not want the initial P. at the end of a line and the surname Neukomm at the beginning of the next line. To prevent this, you should type `P.~Neukomm`.

The following examples show some typical uses:

```
Theorem~\ref{T:main} in Section~\ref{S:intro}
```

```
        the lattice~$L$.

        Sections~\ref{S:modular} and~\ref{S:distributive}

        In~$L$, we find
```

Ellipses The text ellipsis, . . . , is produced using the \dots command. Typing three periods produces ... (notice that the spacing is wrong).

Ligatures Certain groups of characters, when typeset, are joined together—such compound characters are called *ligatures.* There are five ligatures that LaTeX typesets automatically: ff, fi, fl, ffi, and ffl.

If you want to prevent LaTeX from forming a ligature, separate the characters with an empty group { } (officially, with \textcompwordmark). Compare iff with iff, typed as iff and if{}f.

2.3.1   *Accents and symbols in text*

LaTeX provides 15 text accents. Type the command for the accent (\ and a character), followed by the letter (in braces) on which you want the accent placed; see Section A.2.

For example, to get Grätzer György, type Gr\"atzer Gy\"orgy and to get Ö type \"O.

To place an accent on top of an i or a j, you must use the *dotless* version of i and j. These are obtained by the commands \i and \j: \'{\i} typesets as í and \v{\j} typesets as ǰ.

Sections A.1 and A.2 list European characters and text accents available in LaTeX. Section A.4 adds some extra text symbols.

2.3.2   *Logos and useful numbers*

\TeX produces TeX and \LaTeX produces LaTeX.

Remember to type \TeX\␣ or \TeX{} if you need a space after TeX (similarly for the others). A better way to handle this problem is discussed on page 113.

LaTeX also stores some useful numbers:

- \day is the day of the month

- \month is the month of the year

- \year is the current year

You can include these numbers in your document by using the \the command:

```
Year: \the\year; month: \the\month; day: \the\day
```

produces a result such as

Year: 2014; month: 1; day: 11

Of more interest is the \today command, which produces today's date in the form: January 11, 2014. It is often used as the argument of the \date command; see Section 1.3.

2.3.3   Hyphenation

In Section 1.4 we discussed optional hyphens.

Practical Rule ■ Hyphenation specifications
In the preamble, list the words that often need help in a command:

```
\hyphenation{set-up as-so-ciate}
```

All occurrences of the listed words will be hyphenated as specified.

Note that in the \hyphenation command the hyphens are designated by - and not by \-, and that the words are separated by spaces, not by commas.

You must use optional hyphens for words with accented characters, as in

```
Gr\"{a}t\-zer
```

Practical Rule ■ Preventing hyphenation
To *prevent* hyphenation of a word, put it in the argument of a \text command or place it unhyphenated in a \hyphenation command.

For example, type

```
\text{database}
```

if you do not want this instance of database hyphenated, or type

```
\hyphenation{database}
```

if you do not want LaTeX to hyphenate any occurrence of the word anywhere after this command in your document. Typing data\-base overrides the general prohibition for this one instance.

2.4   *Comments and footnotes*

Various parts of your source file do not get typeset like the rest. The two primary examples are comments that do not get typeset at all and footnotes that get typeset at the bottom of the page.

2.4.1   Comments

The % symbol tells LaTeX to ignore the rest of the line. For instance, making a note to look up the proper reference:

```
therefore, a reference to Theorem~15 % check this!
```

The % symbol has many uses. For instance,

```
\documentclass[twocolumn,twoside]{amsart}
```

can be typed with explanations, as

```
\documentclass[twocolumn,%  option for two-column pages
twoside,%  format for two-sided printing
]{amsart}
```

so you can easily comment out some options at a later time.

Practical Tip 21. Some command arguments do not allow any spaces. If you want to break a line within an argument list, you can terminate the line with a %, as shown in the previous example.

Practical Tip 22. The 25% rule

If you want a % sign in text, make sure you type it as \%. Otherwise, % comments out the rest of the line. LaTeX does not produce a warning.

Using % to comment out large blocks of text can be tedious even with block comment. The verbatim package includes the comment environment:

```
\begin{comment}
    ...the commented out text...
\end{comment}
```

Practical Rule ■ The comment **environment**

1. \end{comment} must be at the beginning of a line by itself.

2. There can be no comment within a comment.

The comment environment can be very useful in locating errors.

Practical Tip 23. Suppose you have unbalanced braces in your source file (see Section 2.2.2). Working with a *copy* of your source file, comment out the first half at a safe point (not within an environment!) and typeset. If you still get the same error message, the error is in the second half. If there is no error message, the error is in the first half. Comment out the half that has no error.

Now comment out half of the remaining text and typeset again. Check to see whether the error appears in the first half of the remaining text or the second. Continue applying this method until you narrow down the error to a paragraph that you can inspect visually.

Since the `comment` environment requires the `verbatim` package, you must include the line

```
\usepackage{verbatim}
```

in the preamble of the source file; see Section 1.9.

2.4.2 Footnotes

A footnote is typed as the argument of a `\footnote` command. To illustrate the use of footnotes, I have placed one here.[1] This footnote is typed as

```
\footnote{Footnotes are easy to place.}
```

2.5 Changing font characteristics

Although a document class and its options determine how LaTeX typesets characters, there are occasions when you want control over the shape or size of the font used.

2.5.1 Basic font characteristics

You do not have to be a typesetting expert to recognize the following basic font attributes:

Shape Normal text is typeset:
upright (or *roman*)	as this text
slanted	as this text
italic	as this text
small caps	AS THIS TEXT

Monospaced and proportional Typewriters used *monospaced* fonts, that is, fonts all of whose characters are of the same width. Most text editors display text using a monospaced font. LaTeX calls monospaced fonts *typewriter style.* In this book, such a font is used to represent user input and LaTeX's response, such as "`typewriter style text`". Whereas, normal text is typeset in a *proportional* font, such as "proportional text with ii and mm", in which i is narrow and m is wide.

Serifs A *serif* is a small horizontal (sometimes vertical) stroke used to finish off a vertical stroke of a letter, as on the top and bottom of the letter M. LaTeX's standard serif font is Computer Modern roman, such as "serif text". Fonts without serifs are called *sans serif*, such as "sans serif text". Sans serif fonts are often used for titles or for special emphasis.

Series: weight and width The *series* is the combination of weight and width. A font's *weight* is the thickness of the strokes and the *width* is how wide the characters are. The Computer Modern family includes **bold fonts**.

---

[1]Footnotes are easy to place.

Size Most LaTeX documents are typeset with 10 point text unless otherwise instructed. Larger sizes are used for titles, section titles, and so on. Abstracts and footnotes are normally set in 8-point type.

Font family The collections of all sizes of a font is called a *font family*.

2.5.2   *Document font families*

In a document class, the style designer designates three document font families:

1. *Roman* (upright and serifed) document font family

2. *Sans serif* document font family

3. *Typewriter style* document font family

and picks one of these (for documents, as a rule, the roman document font family) as the *document font family* or *normal family.* In all the examples in this book, the document font family is the roman document font family except for presentations which use sans serif. In standard LaTeX, the three document font families are Computer Modern roman, Computer Modern sans serif, and Computer Modern typewriter.

In this book, the roman document font family is Times, the sans serif document font family is Helvetica, and the typewriter style document font family is Computer Modern typewriter. (Examples are typeset in Computer Modern.)

The document font family (normal family) is the default font. You can always switch back to it with

```
\textnormal{...} or {\normalfont ...}
```

Section A.3.1 lists these two commands and three additional pairs of commands to help you switch among the three basic document font families. It also shows the command pairs for the basic font shapes.

Command pairs

The font-changing commands of Section A.3.1 come in two forms:

- A command with an argument, such as \textrm{...}, changes its argument. These are short commands, i.e., they cannot contain a blank line or a \par command.

- A command declaration, such as \rmfamily, carries out the font change following the command and within its scope; see Section 2.2.2.

Practical Tip 24. You should always use commands with arguments for small changes within a paragraph, because you are less likely to forget to change back to the normal font and do not have to worry about italic corrections; see Section 2.5.4.

For font changes involving more than one paragraph, use command declarations.

2.5.3   Shape commands

There are five pairs of commands to change the font shape:

- `\textup{...}` or `{\upshape ...}` switch to the upright shape
- `\textit{...}` or `{\itshape ...}` switch to the *italic shape*
- `\textsl{...}` or `{\slshape ...}` switch to the *slanted shape*
- `\textsc{...}` or `{\scshape ...}` switch to SMALL CAPITALS
- `\emph{...}` or `{\em ...}` switch to *emphasis*

The document class specifies how emphasis is typeset. As a rule, it is italic or slanted unless the surrounding text is italic or slanted, in which case it is upright. For instance,

`\emph{Rubin space}`

in the statement of a theorem is typeset as

the space satisfies all three conditions, a so-called Rubin space *that ...*

The emphasis changed the style of Rubin space from italic to upright.

💡 **Practical Tip 25.** Be careful not to interchange the command pairs. For instance, if by mistake you type `{\textit serif}`, the result is *s*erif. Only the *s* is italicized since `\textit` takes `s` as its argument.

Practical Rule ■ Abbreviations and acronyms

For abbreviations and acronyms use small caps, except for two-letter geographical acronyms.

So Submitted to TUG should be typed as

`Submitted to \textsc{tug}`

Note that only the lowercase characters in the argument of the `\textsc` command are printed as small caps. `\textsc{TUG}` prints as TUG, not as TUG.

2.5.4   Italic corrections

The phrase

when using a *serif* font

can be typed as follows:

`when using a {\itshape serif\/} font`

The \/ command before the closing brace is called an *italic correction.* Notice that {\itshape M}M typesets as *M*M, where the *M* is leaning into the M. Type {\itshape M\/}M to get the correct spacing *M*M. Compare the typeset phrase from the previous example with and without an italic correction:

when using a *serif* font
when using a *serif* font

The shape commands with arguments do not require italic correction. The corrections are provided automatically where needed. Thus you can type the phrase when using a *serif* font the easy way:

```
when using a \textit{serif} font
```

💡 **Practical Tip 26.** Whenever possible, let LaTeX take care of the italic correction.

2.5.5   Series

These attributes play a very limited role with the Computer Modern fonts. There is only one important pair of commands,

```
\textbf{...}    {\bfseries ...}
```

to change the font to bold.

2.5.6   Size changes

LaTeX documents, as a rule, are typeset in 10 point type. The 11 point and 12 point type are often used for greater readability and some journals require 12 point—if this is the case, use the 12pt document class option; see Section 6.6.5. The sizes of titles, subscripts, and superscripts are automatically set by the document class, in accordance with the font size option.

If you must change the font size for some text—it is seldom necessary to do so in a document—the following command declarations are provided:

```
\Tiny  \tiny  \SMALL  \Small  \small
              \normalsize
\large  \Large  \LARGE  \huge  \Huge
```

See Section A.3.2 for a visual representation of these commands.
The command \SMALL is also called \scriptsize and the command \Small is also called \footnotesize. The font size commands are listed in order of increasing—to be more precise, nondecreasing—size.

2.6   Lines, paragraphs, and pages

When typesetting a document, LaTeX breaks the text into lines, paragraphs, and pages. Sometimes you may not like how LaTeX has chosen to lay out your text. There are ways to influence how LaTeX does its work and these are discussed in this section.

2.6.1   Lines

LaTeX typesets a document one paragraph at a time. It tries to split the paragraph into lines of equal width with balanced spacing. If it fails to do so successfully and a line is too wide, you get an `overfull \hbox` message, as discussed in Section 1.4.

Breaking lines

There are two forms of the line breaking command:

- The `\\` and `\newline` commands break the line at the point of insertion but do not stretch it.

- The `\linebreak` command breaks the line at the point of insertion and stretches the line to make it of the normal width.

The text following any of these commands starts at the beginning of the next line, without indentation. The `\\` command is often used, but `\linebreak` is rarely seen. The `\\` command has an important variant: `\\[length]`, where `length` is the interline space you wish to specify after the line break. For instance, `length` may be `12pt`, `.5in`, or `1.2cm`. Note how the units are abbreviated.

2.6.2   Paragraphs

Paragraphs are separated by blank lines or by the `\par` command. Error messages always show paragraph breaks as `\par`.

Indentation can be prevented with the `\noindent` command and can be forced with the `\indent` command.

2.6.3   Pages

There are two page breaking commands:

- `\newpage`, which breaks the page at the end of the line next completed but does not stretch it

- `\pagebreak`, which breaks the page at the point of insertion and stretches it to normal length

Text following either command starts at the beginning of the next page, indented.

2.7   Spaces

The judicious use of horizontal and vertical space is an important part of the formatting of a document. Fortunately, most of the spacing decisions are made by the document class, but LaTeX has a large number of commands that allow the user to insert horizontal and vertical spacing.

Remember that LaTeX ignores excess spaces, tabs, and end-of-line characters in the source file. If you need to add horizontal or vertical space in the typeset file, then you must choose from the commands in this section.

Practical Tip 27. Use them sparingly!

2.7.1   Horizontal spaces

When typing text, there are four commands that are often used to create horizontal space; three are shown between the bars in the display below:

```
\,          ⊔
\quad       ⊔__⊔
\qquad      ⊔____⊔
```

The fourth is the `\,` command, producing a thin space. You have seen its first example in this book at the end of Section .

The `\hspace` command takes a length as a parameter. For example, `\textbar\hspace{12pt}\textbar` prints as | |. This command is ignored at the beginning of a line; use `\hspace*` instead.

The length can be negative. This is often used when placing illustrations. Negative thin space is provided by the `\!` command.

The `\hfill`, `\dotfill`, and `\hrulefill` commands fill all available space in the line with spaces, dots, or a horizontal line, respectively. If there are two of these commands on the same line, the space is divided equally between them. These commands can be used to center text, to fill lines with dots in a table of contents, and so on.

To obtain

```
   2. Boxes.....................................................34
ABC                           and                          DEF
ABC_____and_____DEF
```

type

```
2. Boxes\dotfill 34\\
ABC\hfill and\hfill DEF\\
ABC\hrulefill and\hrulefill DEF
```

In a centered environment—such as the `center` environment—you can use `\hfill` to set a line flush right:

```
                    This is the title
                                                  First Draft
                         Author
```

typed as

```
\begin{center}
  This is the title\\
  \hfill First Draft\\
  Author
\end{center}
```

2.7.2   Vertical spaces

You can add some interline space with the command \\[length], as discussed in Section 2.6.1. You can also do it with the \vspace command, which works just like the \hspace command; see Section 2.7.1. Here are some examples:

```
\vspace{12pt}  \vspace{.5in}  \vspace{1.5cm}
```

This command is ignored at the beginning of a page; use \vspace* instead.

Standard amounts of vertical space are provided by the three commands

```
\smallskip  \medskip  \bigskip
```

As a rule, they represent a vertical space of 3 points, 6 points, and 12 points, respectively.

The vertical analogue of \hfill is \vfill. This command fills the page with vertical space so that the text before the command and the text after the command stretch to the upper and lower margin.

2.8   Boxes

Sometimes it can be useful to typeset text in an imaginary box, and treat that box as a single large character. A single-line box can be created with the \text command.

2.8.1   Line boxes

The \text command provides a *line box* that typesets its argument without line breaks. As a result, you may find the argument extending into the margin. The resulting box is handled by LaTeX as if it were a single large character. For instance,

```
\text{database}
```

causes LaTeX to treat the eight characters of the word database as if they were one. This technique has two major uses. It prevents LaTeX from hyphenating the argument; see Section 2.3.3. It allows you to use the phrase in the argument in a formula; see Section 4.3.3.

The argument of \text is typeset in a size appropriate for its use, for example, as a subscript or superscript. See Section 4.3.3 for an example.

2.8.2   Marginal comments

Do not
use this
much.

The \marginpar command allows you to add marginal comments. So

    \marginpar{Do not use this much.}

produces the comment displayed in the margin. Marginal comments appear on the left
on even-numbered pages and on the right on odd-numbered pages.

2.8.3   Paragraph alignments

Horizontal alignment of a paragraph is controlled by the flushleft, flushright,
and center environments. Within the flushright and center environments, it is
customary to force new lines with the \\ command, while in the flushleft environ-
ment, you normally allow LaTeX to wrap the lines.

There are command declarations that correspond to these environments:

- \centering centers text

- \raggedright left aligns text

- \raggedleft right aligns text

The effect of one of these commands is almost the same as that of the corresponding
environment except that the environment places additional vertical space before and
after the displayed paragraphs. For such a command declaration to affect the way
a paragraph is formatted, the scope must include the whole paragraph, including the
blank line at the end of the paragraph, preferably indicated with a \par command.

3

Text environments

We start, in Section 3.1, by discussing blank lines in displayed text environments. Then we proceed in Section 3.2 to the often used displayed text environments: lists. The most important displayed text environments in math are proclamations (also called, theorem-like structures), proclamations with style, and the `proof` environment, discussed in Sections 3.3 and 3.4.

3.1 Blank lines in displayed text environments

Practical Rule ■ **Blank lines in displayed text environments**

1. Blank lines are ignored immediately after `\begin{name}` except in a `verbatim` environment;

2. blank lines are ignored immediately before `\end{name}` except in a `verbatim` environment or `proof` environment.

3. A blank line after \end{name} forces the text that follows to start a new paragraph.

4. As a rule, you should not have a blank line before \begin{name}.

5. The line after any theorem or proof always begins a new paragraph, even if there is no blank line or \par command.

3.2   List environments

LaTeX provides three list environments: enumerate, itemize, and description.

Most document classes redefine the spacing and some stylistic details of lists, especially since the list environments in the legacy document classes are not very pleasing. In this section, the list environments are formatted as they are by our standard document class, amsart. Throughout the rest of the book, lists are formatted as specified by this book's designer.

3.2.1   Numbered lists

A *numbered list* is created with the enumerate environment:

1. Half-smooth Hausdorff;

2. Metrizably smooth.

typed as

```
\begin{enumerate}
   \item Half-smooth Hausdorff\label{Hausdorff};
   \item Metrizably smooth\label{smooth}.
\end{enumerate}
```

Each item is introduced with an \item command. The numbers LaTeX generates can be labeled and cross-referenced; see Section 1.8.2. This construct can be used in theorems and definitions, for listing conditions or conclusions.

If you use \item in the form \item[], you get an unnumbered item in the list.

Practical Tip 28. A single unnumbered item will display the text indented on the left.

Use the quote environment to display text indented on both sides as in the last but one paragraph of page 1.

3.2.2   Bulleted lists

A *bulleted list* is created with the itemize environment:

⌐

- To introduce the concept of smooth functions.

- To show their usefulness in differentiation.

∟

typed as

```
\begin{itemize}
   \item To introduce the concept of smooth functions.
   \item To show their usefulness in differentiation.
\end{itemize}
```

3.2.3   Captioned lists

In a *captioned list* each item has a title (caption) specified by the optional argument of the \item command. Such lists are created with the description environment:

⌐

Chopped lattice a reduced form of a lattice.

Boolean triples a powerful lattice construction.

∟

typed as

```
\begin{description}
  \item[Chopped lattice] a reduced form of a lattice.
  \item[Boolean triples] a powerful lattice construction.
\end{description}
```

3.2.4   A rule and combinations

There is only one rule you must remember.

Practical Rule ■ **List environments**

An \item command must immediately follow
\begin{enumerate}, \begin{itemize}, or \begin{description}.

If you break this rule, you get an error message. For instance,

```
\begin{description}
This is wrong!
  \item[Chopped lattice] a reduced lattice;
```

gives the error message

```
! LaTeX Error: Something's wrong--perhaps a missing \item.

l.105    \item[Chopped lattice]
                                 a reduced lattice;
```

If you see this error message, remember the rule for list environments and check for text preceding the first \item.

You can nest up to four list environments.

Practical Tip 29. If the text following an \item command starts with an opening square bracket, [, then LaTeX thinks that \item has an optional argument. To prevent this problem from occurring, type [as {[}. Similarly, a closing square bracket,], *inside* an optional argument should be typed as {]}.

Practical Tip 30. You may want to use a list environment solely for the way the items are displayed, without any labels. You can achieve this effect by using \item[].

You can change the style of the numbers in an enumerate environment using David Carlisle's enumerate package.

3.3   Proclamations (theorem-like structures)

Theorems, lemmas, definitions, and so forth are a major part of mathematical writing. In LaTeX, these constructs are typed in displayed text environments called *proclamations* or *theorem-like structures*.

There are two steps required for a proclamations:

Step 1 *Define* the proclamation with a \newtheorem command *in the preamble* of the document. For instance, the line

```
\newtheorem{theorem}{Theorem}
```

defines a theorem environment.

Step 2 *Invoke* the proclamation as an environment *in the body* of your document. Using the proclamation definition from Step 1, type

```
\begin{theorem}
   My first theorem.
\end{theorem}
```

to produce a theorem:

Theorem 1. *My first theorem.*

In the proclamation definition

```
\newtheorem{theorem}{Theorem}
```

the first argument, `theorem`, is the name of the environment that invokes the theorem. The second argument, `Theorem`, is the name that is used when the proclamation is typeset. LaTeX numbers the theorems automatically and typesets them with vertical space above and below. The phrase **Theorem 1.** appears, followed by the theorem itself, which can be emphasized; see Section 3.3.1.

You can also specify an optional argument,

```
\begin{theorem}[The Fuchs-Schmidt Theorem]
    The statement of the theorem.
\end{theorem}
```

that appears as the name of the theorem:

Theorem 1 (The Fuchs-Schmidt Theorem). *The statement of the theorem.*

Practical Tip 31. Lists in proclamations
If a proclamation starts with a list environment, precede the list by `\hfill`.

If you do not, as in

```
\begin{definition}\label{D:prime}
    \begin{enumerate}
    \item $u$ is \emph{bold} if $u = x^2$.\label{mi1}
    \item $u$ is \emph{thin} if $u = \sqrt{x}$.\label{mi2}
    \end{enumerate}
\end{definition}
```

(using `\theoremstyle{definition}` for the `definition` proclamation) your type-set list starts on the first line of the proclamation:

Definition 1.   (1) u *is* bold *if* $u = x^2$.
          (2) u *is* thin *if* $u = \sqrt{x}$.

If you add the `\hfill` command,

```
\begin{definition}\hfill
\begin{enumerate}
```

the list in the definition typesets correctly (note: bold and thin are upright!):

Definition 1.   (1) u *is* bold *if* $u = x^2$.
          (2) u *is* thin *if* $u = \sqrt{x}$.

Consecutive numbering

If you want to number two sets of proclamations consecutively, you can do so by first defining one proclamation, and then using its name as an optional argument of the second proclamation. For example, to number the lemmas and propositions in your paper consecutively, you type the following two lines in your preamble:

```
\newtheorem{lemma}{Lemma}
\newtheorem{proposition}[lemma]{Proposition}
```

Lemmas and propositions are then consecutively numbered as **Lemma 1**, **Proposition 2**, **Proposition 3**, and so on.

Numbering within a section

The \newtheorem command can also have another optional argument; it causes LaTeX to number the proclamations within sections. For example,

```
\newtheorem{lemma}{Lemma}[section]
```

numbers the lemmas in Section 1 as **Lemma 1.1** and **Lemma 1.2**. In Section 2, you have **Lemma 2.1** and **Lemma 2.2**, and so on.

3.3.1   *Proclamations with style*

You can choose one of three styles for your proclamations by preceding the definitions with the \theoremstyle{style} command, where *style* is one of the following:

- plain, emphasized text with space above and below

- definition, upright text with space above and below

- remark, upright text with no extra space

    For instance,

```
\theoremstyle{plain}
\newtheorem{theorem}{Theorem}
\newtheorem{corollary}{Corollary}
\newtheorem*{main}{Main Theorem}
\newtheorem{lemma}{Lemma}
\newtheorem{proposition}{Proposition}

\theoremstyle{definition}
\newtheorem{definition}{Definition}

\theoremstyle{remark}
\newtheorem*{notation}{Notation}
```

The third and the last proclamations use the `\newtheorem*` command, the unnumbered version of `\newtheorem`.

3.4   Proof environments

A proof is placed in a `proof` environment. For instance,

Proof. This is a proof, delimited by the q.e.d. symbol.   □

typed as
```
\begin{proof}
This is a proof, delimited by the q.e.d.\ symbol.
\end{proof}
```
The end of the proof is marked with the symbol □ at the end of the line.

Practical Tip 32. Lists in proofs

If a proof starts with a list environment, precede the list by `\hfill`.
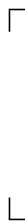
To substitute another phrase for *Proof,* such as *Necessity,* as in

Necessity. This is the proof of necessity.   □

use the `proof` environment with an optional argument:
```
\begin{proof}[Necessity]
This is the proof of necessity.
\end{proof}
```
There is a problem with the placement of the q.e.d. symbol if the proof ends with a displayed formula or with a list environment. For instance,
```
\begin{proof}
Now the proof follows from the equation
\[ a^2 = b^2 + c^2. \]
\end{proof}
```
typesets as

Proof. Now the proof follows from the equation

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

□

To correct the placement of the q.e.d. symbol, use the `\qedhere` command:

```
\begin{proof}
Now the proof follows from the equation
\[ a^2 = b^2 + c^2.\qedhere \]
\end{proof}
```

which typesets as

⌐

Proof. Now the proof follows from the equation

$$a^2 = b^2 + c^2. \quad \square$$

└

3.5   Tabular environments

A `tabular` environment creates a table that LaTeX treats as a "large symbol". In particular, a table cannot be broken across pages.

Here is a simple table,

Name	1	2	3
Peter	2.45	34.12	1.00
John	0.00	12.89	3.71

, typeset inline. This looks awful, but it does make the point that the table is just a "large symbol". The table is typed as

```
\begin{tabular}{ | l | r | r | r | }
   \hline
   Name    & 1    & 2    & 3   \\ \hline
   Peter   &  2.45 & 34.12 & 1.00\\ \hline
   John    &  0.00 & 12.89 & 3.71\\ \hline
   \end{tabular}
```

with no blank line before or after the environment.

This table can be horizontally centered with a `center` environment, which pads it with space before and after, or with the `\centering` command, which centers but adds no space. It can also be placed within a `table` environment, which makes the table "float"; see Section 6.4.2.

```
\begin{table}
   \centering
     \begin{tabular}{ | l | r | r | r | }
       \hline
       Name    & 1    & 2    & 3   \\ \hline
       Peter   &  2.45 & 34.12 & 1.00\\ \hline
       John    &  0.00 & 12.89 & 3.71\\ \hline
     \end{tabular}
     \caption{Tabular table}\label{Ta:first}
\end{table}
```

Name	1	2	3
Peter	2.45	34.12	1.00
John	0.00	12.89	3.71

Table 3.1: Tabular table.

This table is displayed as Table 3.1.

The table number can be referenced using the command `\ref{Ta:first}`. Note that the label must be typed *after* the caption and *before* the `\end{table}` command.

Practical Rule ■ `tabular` **environments**

1. `\begin{tabular}` requires an argument consisting of a character `l`, `r`, or `c`, meaning left, right, or center alignment, for each column, and optionally, the | symbols. Each | indicates a vertical line in the typeset table. Spaces in the argument are ignored but can be used for readability.

2. Columns are separated by ampersands (`&`) and rows are separated by `\\`.

3. `&` absorbs spaces on either side.

4. The `\hline` command creates a horizontal rule in the typeset table. It is placed either at the beginning of the table (after the `\begin` line) or it must follow a `\\` command.

5. If you use a horizontal line to finish the table, you must separate the last row of the table from the `\hline` command with the `\\` command.

6. `\begin{tabular}` takes an optional argument, `b` or `t`, to specify the bottom or the top vertical alignment of the table with the baseline. The default is center alignment.

More column-formatting commands

The required argument of the `tabular` environment can contain column-formatting commands of various types.

An @-*expression,* for instance, `@{.}`, replaces the space LaTeX normally inserts between two columns with its argument. For example,

```
\begin{tabular}{r @{.} l}
   3&78\\
   4&261\\
   4
\end{tabular}
```

creates a table with two columns separated by a decimal point. In effect, you get a single, decimal-aligned column:

⌐
      3.78
      4.261
      4.
∟

This example is an illustration. You should use David Carlisle's `dcolumn` package if you need a decimal-aligned column.

The width of a column depends on the entries in the column by default. You can specify a width by using the p column specifier:

```
p{width}
```

For instance, if you want the first column of Table 3.1 to be 1 inch wide, then type

```
\begin{tabular}{ | p{1in} | r | r | r | }\hline
   Name    & 1    & 2     & 3   \\ \hline
   Peter   & 2.45 & 34.12 & 1.00\\ \hline
   John    & 0.00 & 12.89 & 3.71\\ \hline
   David   & 2.00 & 1.85  & 0.71\\ \hline
\end{tabular}
```

which typesets as

Name	1	2	3
Peter	2.45	34.12	1.00
John	0.00	12.89	3.71
David	2.00	1.85	0.71

To center the items in the first column, precede *each* item with a \centering command.

MiL4 has lots more on tables.

4

Inline formulas

LaTeX was designed for typesetting formulas. I address this topic in detail.

As we discussed in Section 1.5, a formula can be typeset *inline*, as part of the current paragraph, or *displayed*, on a separate line or lines with vertical space before and after the formula. In this chapter, we discuss formulas that are set inline. In Chapter 5 we address displayed formulas.

We start with a discussion of LaTeX's basic math environments (Section 4.1) and spacing rules in math (Section 4.2). We started discussing the basic constructs of a formula in Section 1.7; these will be discussed in more detail in Section 4.3.

Delimiters, operators, and math accents are dealt with in Sections 4.4–4.6. In Section 4.7, we discuss three types of stretchable horizontal lines that can be used above or below a formula: braces, bars, and arrows. There are also stretchable arrow math symbols.

LaTeX has a very large number of math symbols. Section 4.8 classifies and describes them. Section 4.9 discusses how to build new symbols from existing ones. Math alphabets and symbols are discussed in Section 4.10.

4.1   Formula environments

As we have seen in Section 1.5, a formula in a LaTeX document can be typeset *inline,* like the congruence $a \equiv b \ (\theta)$ or the integral $\int_{-\infty}^{\infty} e^{-x^2} \, dx = \sqrt{\pi}$, or *displayed*, as in

$$a \equiv b \quad (\theta)$$

and

$$\int_{-\infty}^{\infty} e^{-x^2} \, dx = \sqrt{\pi}$$

Notice how changing these two formulas from inline to displayed affects their appearance. We discuss displayed formulas in Chapter 5.

Practical Rule ■ Formulas

No blank lines are permitted in a formula.

If you violate this rule, LaTeX generates an error message,

```
! Missing $ inserted.
<inserted text>
                $
...
l.117
```

where the line number points inside the environment.

4.2   Spacing rules

In text, the most important spacing rule is that any number of spaces in the source file equals one space in the typeset document. The spacing rule for a formula is even more straightforward:

Practical Rule ■ Spacing in math

LaTeX ignores spaces in math.

In other words, all spacing in a formula is provided by LaTeX. For instance, `$a+b=c$` and `$a + b = c$` are both typeset as $a + b = c$.

There are two exceptions to this rule:

1. A space indicating the end of a command name is recognized. For instance, in `$a \quad b$` LaTeX does not ignore the space between `\quad` and `b`.

2. If you switch back to text inside a formula with a `\text` command (see Sections 1.7, 4.3.3), then the text spacing rules apply in the argument of such a command.

LaTeX provides controls for spaces in typeset math. The spaces you type in math do not affect the typeset document. But keep this tip in mind.

Practical Tip 33. Format your source file so that it is easy to read.

When typing a source file, the following is good practice:

- Leave spaces before and after binary operations and binary relations, including the equal sign.

- Indent—by three spaces, for example—the contents of environments so they stand out.

- Keep a formula on a single line of the source file, if it fits.

Develop your own style of typing math, and stick with it.

Practical Tip 34. The spacing after a comma is different in math and text.

Practical Tip 35. Do not leave a trailing comma or period (any punctuation) in inline formulas.

So do not type

```
If $a = b,$ then
```

but move the comma out. (Of course, punctuation marks are within for displayed formulas.)

4.3   Basic constructs

As we discussed in Section 1.7, a formula (inline or displayed) is built up by combining various building blocks. In this section, we look at some enhancements. Read carefully the basic constructs *important for your work*.

Some of these constructs produce tall formulas so they are mostly used in displayed formulas. Only the shorter ones are appropriate for inline formulas. We'll list them here anyway, but warn you many times: Remember Rule 8 for formulas in Section 1.13.

4.3.1   Integrals

You have already seen the formula $\int_{-\infty}^{\infty} e^{-x^2} \, dx = \sqrt{\pi}$ in both inline and displayed forms in Section 1.7. The lower limit is typeset as a subscript and the upper limit is typeset as a superscript. To force the limits below and above the integral symbol,

use the \limits command. The \nolimits command does the reverse. To typeset $\int\limits_{-\infty}^{\infty} e^{-x^2} \, dx = \sqrt{\pi}$, type

`$\int\limits_{-\infty}^{\infty} e^{-x^{2}} \, dx = \sqrt{\pi}$`

There are five commands to produce variants of the basic integral symbol:

\oint   \iint   \iiint   \iiiint   \idotsint

which typeset as

$$\oint \qquad \iint \qquad \iiint \qquad \iiiint \qquad \int\cdots\int$$

For complicated bounds, use the \substack command or the subarray environment; see Section 4.5.4.

4.3.2 Roots

In $\sqrt[g]{5}$, typed as `$\sqrt[g]{5}$`, the placement of g is not very pleasing. Two additional commands are provided by LaTeX to adjust the position of g:

\leftroot moves g *left*—or *right* with a negative argument

\uproot moves g *up*—or *down* with a negative argument

You may prefer to see $\sqrt[g]{5}$, typed as `$\sqrt[\leftroot{2} \uproot{2} g]{5}$`.
    Note that LaTeX is very finicky with this optional argument. Typing a space after the symbol [, gives the error message

`! Package amsmath Error: Invalid use of \uproot.`

4.3.3 Text in math

Note that the argument of the \text command is always typeset in a single line.
    Sometimes it is more convenient to type a formula within the argument of a \text command rather than end the \text and start another, as in $A = \{\, x \mid \text{for } x \text{ large} \,\}$ which is typed as

`$A = \{\, x \mid \text{for $x$ large} \,\} $`

The \text command correctly sizes its argument to match the context. The formula $a_{\text{left}} + 2 = a_{\text{right}}$ is typed as

`$a_{\text{left}} + 2 = a_{\text{right}}$`

Note that \text typesets its argument *in the size and shape* of the surrounding text. If you want the text in a formula to be typeset in the document font family (see Section 2.5.2) independent of the surrounding text, use

`\textnormal{ ... }`

For instance, if you have a constant a_{right}, typed as `$a_{\text{right}}$`, then in a theorem:

Theorem 1. *The constant a_{right} is recursive in a.*

the subscript is wrong. Type the constant as `$a_{\textnormal{right}}$` to get it right.

4.4 Delimiters

Delimiters were introduced in Section 1.7. All the standard delimiters are listed in Section B.6.

Note that delimiters are math symbols with special spacing rules. You can use them in any way you please, not only in pairs. LATEX does not stop you from typing `\uparrow(x)]`, which typesets as $\uparrow (x)]$.

Observe the difference in spacing between $||a||$ and $\|a\|$. The first, $||a||$, was typed incorrectly as `$|| a ||$`. As a result, the vertical bars are too far apart. The second was typed correctly using the appropriate delimiter commands: `$\| a \|$`. Here they are again side-by-side, enlarged: $\big|\big|a\big|\big|$ $\big\|a\big\|$.

4.4.1 Stretching delimiters

All delimiters, except the four "corners" (the last four delimiters in Section B.6), can stretch to enclose the formula. This formula $\left(\frac{1}{2}\right)^{\alpha}$ is typed as

`$\left( \frac{1}{2} \right)^{\alpha}$`

The `\left` and `\right` commands instruct LATEX to stretch the parentheses.

The general construction is

`\left delim1 and \right delim2`

where `delim1` and `delim2` are chosen from the listing in Section B.6. They are usually, but not always, a matching pair—see the examples below. LATEX inspects the formula between the `\left` and `\right` commands and decides what size delimiters to use. The `\left` and `\right` commands *must be paired* in order for LATEX to know the extent of the material to be vertically measured. However, the delimiters need not be the same.

If you want to stretch a single delimiter, you have to pair it with a *blank delimiter*, represented by the `\left.` and `\right.` commands.

Here are some examples of stretching delimiters (some not stretching in these examples):

$$\left|\tfrac{a+b}{2}\right|, \quad \left\|A^2\right\|, \quad \left(\tfrac{a}{2}, b\right], \quad F(x)\big|_a^b$$

typed as

```
$\left| \frac{a + b}{2} \right|, \quad \left\| A^{2} \right\|,
   \quad \left( \frac{a}{2}, b \right],
   \quad \left. F(x) \right|_{a}^{b}$
```

Recall Rule 8 for formulas in Section 1.13.

There are also two convenient abbreviations:

```
\left<    for   \left\langle
\right>  for   \right\rangle
```

The \left and \right commands have one more use. For the delimiters |, \|, and all the arrows, the same symbol represents the left and right delimiters, which can sometimes cause problems. In such cases, you should use the \left and \right commands to tell LaTeX whether the delimiter is a left or a right delimiter. LaTeX also provides the \lvert and \rvert for | as left and right delimiter, and \lVert and \rVert for \|.

4.4.2   Delimiters that do not stretch

LaTeX provides the \big, \Big, \bigg, and \Bigg commands to produce delimiters of larger sizes. These delimiters do not stretch. For example,

```
$(\quad \big(\quad \Big(\quad \bigg(\quad \Bigg($
```

typesets as

$$(\quad \Big(\quad \bigg(\quad \Bigg(\quad \left($$

LaTeX also provides the more specific

```
\bigl, \Bigl, \biggl, \Biggl, \bigr, \Bigr, \biggr, and \Biggr
```

commands to produce larger left and right delimiters.

For integral evaluation, you can choose one of the following:

$$F(x)|_a^b \quad F(x)\big|_a^b \quad F(x)\bigg|_a^b$$

typed as

```
\[ F(x) |^{b}_{a} \quad F(x) \bigr|^{b}_{a}  \quad
   F(x) \Bigr|^{b}_{a} \]
```

In a number of situations the stretching done by LaTeX is not ideal, so you should use a non-stretching variant. Remember Rule 8 for formulas in Section 1.13.

4.4.3   Delimiters as binary relations

The symbol $|$ can be used as a delimiter, as in $|x + y|$, and also as a binary relation, as in $\{\, x \in \mathcal{R} \mid x^2 \leq 2 \,\}$. As a binary relation it is typed as \mid. The previous formula is typed as

`$\{\, x \in \mathcal{R} \mid x^{2} \leq 2 \,\}$`

\bigm | and \biggm | produce larger variants: $\big|$ and $\bigg|$. (\bigm \mid does not work.)

4.5   Operators

4.5.1   Types of operators

There are two types of operators:

1. *Operators without limits*, such as \sin

2. *Operators with limits*, such as \lim, that take a subscript in inline mode and a "limit" when displayed as a formula. For example, $\lim_{x\to 0} f(x) = 1$ is typed as

   `$\lim_{x \to 0}  f(x) = 1$`

   The operators are listed in two tables in Section B.6.   Here are some examples: $\varliminf_{x\to 0} \quad \varlimsup_{x\to 0}$ which are typed as

`$\varliminf_{x \to 0} \quad  \varlimsup_{x \to 0}$`

4.5.2   Congruences

In addition to \pmod and \pod, there are two more variants: \mod and \bmod, illustrated by $a \equiv v \mod \theta$ and $a \bmod b$ typed as $a \equiv v \mod{\theta}$ and $a \bmod b$.

4.5.3   Large operators

Here is a sum typeset inline, $\sum_{i=1}^{n} x_i^2$, and displayed,

$$\sum_{i=1}^{n} x_i^2$$

In the latter form, the sum symbol is larger. Operators that behave in this way are called *large operators.* Section B.7.1 gives a complete list of large operators.

You can use the \nolimits command if you wish to show the limits of large operators as subscripts and superscripts in a displayed math environment.

The formula $\bigsqcup_{\mathfrak{m}} X = a$ is typed as

`$\bigsqcup\nolimits_{ \mathfrak{m} } X = a$`

You can use the \limits command if you wish to show the limits of large operators below and above the operator symbol in an inline math environment. For example, $\bigsqcup\limits_{\mathrm{m}} X = a$ is typed as `$\bigsqcup\limits_{ \mathfrak{m} } X = a$`.

4.5.4   Multiline subscripts and superscripts

The \substack command provides multiline limits for large operators. For instance, $\sum_{\substack{i<n \\ i\text{ even}}} x_i^2$ is typed as

`$\sum_{ \substack{ i < n\\ i \text{ even} } } x_{i}^{2}$`

It is much easier to read displayed:

$$\sum_{\substack{i<n \\ i\text{ even}}} x_i^2$$

There is only one rule to remember. Use the line separator command \\. You can use the \substack command wherever subscripts or superscripts are used.

The lines are centered by \substack, so if you want them set flush left, as in

$$\sum_{\substack{i<n \\ i\text{ even}}} x_i^2$$

then use the subarray environment with the argument l:

```
\[ \sum_{ \begin{subarray}{l}
             i < n\\
             i \text{ even}
          \end{subarray} }
    x_{i}^{2} \]
```

4.6   Math accents

The accents used in text (see Sections 1.2 and 2.3.1) cannot be used in math formulas. For accents in formulas a separate set of commands is provided. All math accents are shown in Section B.8. The amsxtra package is needed for the accents in the second column. To use them, make sure to place the line

\usepackage{amsxtra}

in the preamble. For instance, `$a\spbreve$` typesets as $a^{\smallsmile}$.

You can also use double accents, such as `\[ \hat{\hat{A}} \]` which typesets as $\hat{\hat{A}}$.

The two "wide" varieties, \widehat and \widetilde, expand to fit the symbols (their arguments) covered: $\widehat{A}$, $\widehat{iiiii}$, and $\widetilde{A}$, $\widetilde{iiiii}$ (the last example is typed as `$\widetilde{iiiii}$`). If the base is too wide, the accent is centered: $\widehat{ABCDE}$.

Notice the difference between $\bar{a}$ and $\overline{a}$, typed as

`$\bar{a}$ $\overline{a}$`

For other examples of the \overline command, see Section 4.7.2.

4.7 Stretchable horizontal lines

LaTeX provides three types of stretchable horizontal lines that appear above or below a formula: braces, bars, and arrows. There are also stretchable arrow symbols. Recall Rule 8 for formulas in Section 1.13!

4.7.1 Horizontal braces

The \overbrace command places a brace of variable size above its argument, as in

$$\overbrace{a + b + \cdots + z}$$

which is typed as
\[\overbrace{a + b + \dots + z} \]

A superscript adds a label to the brace, as in

$$\overbrace{a + a + \cdots + a}^{n}$$

which is typed as
\[\overbrace{a + a + \dots + a}^{n} \]

The \underbrace command works similarly, placing a brace below its argument. A subscript adds a label to the brace. Avoid these inline, see Rule 8 for formulas in Section 1.13!

4.7.2 Overlines and underlines

The \overline and \underline commands draw lines above or below a formula. For example,

$$\overline{\overline{X} \cup \overline{\overline{X}}} = \overline{\overline{X}}$$

is typed as
\[\overline{ \overline{X} \cup \overline{\overline{X}}} }
    = \overline{ \overline{X} } \]

Similarly, you can place arrows above and below an expression:

$$\overleftarrow{a} \quad \overrightarrow{aa} \quad \overleftrightarrow{aaa}$$

$$\underleftarrow{aaaa} \quad \underrightarrow{aaaaa} \quad \underleftrightarrow{aaaaaa}$$

which is typed as (see Section 5.1.1 for the gather* environment):
\begin{gather*}
\overleftarrow{a} \quad \overrightarrow{aa}
    \quad \overleftrightarrow{aaa}\\
\underleftarrow{aaaa} \quad \underrightarrow{aaaaa}
\quad \underleftrightarrow{aaaaaa}
\end{gather*}

4.8   Spacing of symbols

LaTeX provides a large variety of math symbols: Greek characters (α), binary operations ($\circ$), binary relations ($\leq$), negated binary relations ($\nleq$), arrows ($\nearrow$), delimiters ($\{$), and so on. All the math symbols provided by LaTeX are listed in the tables of Appendix B. (Some of these symbols need the amssymb package.)

Consider the formula

$$A = \{x \in X \mid x\beta \geq xy > (x + 1)^2 - \alpha\}$$

which is typed as

`\[ A = \{x \in X \mid x \beta \geq xy > (x + 1)^{2} - \alpha\} \]`

The spacing of the symbols in the formula varies. In $x\beta$, the two symbols are very close. In $x \in X$, there is some space around the $\in$, and in $x + 1$, there is somewhat less space around the $+$.

4.8.1   Classification

LaTeX classifies symbols into several categories or *types* and spaces them accordingly. In the formula $A = \{x \in X \mid x\beta \geq xy > (x + 1)^2 - \alpha\}$ we find

- Ordinary math symbols: A, x, X, β, and so on

- Binary relations: $=$, $\in$, $\mid$, $\leq$, $\subseteq$, and $\prec$

- Binary operations: $+$ and $-$

- Delimiters: $\{$, $\}$, $($, and $)$

As a rule, you do not have to be concerned with whether or not a given symbol in a formula, say $\times$, is a binary operation. LaTeX spaces the typeset symbol correctly. However, in the formula $\prec \subseteq \leq$, the binary relations $\prec$ and $\leq$ are math symbols. So type `\mathord{\prec} \subseteq \mathord{\leq}` or `{\prec}\subseteq {\leq}`. Typing `\prec \subseteq \leq`, you get $\prec\subseteq\leq$.

4.8.2   Three exceptions

There are three symbols with more than one classification: $+$, $-$, and $\mid$.

$+$ or $-$ could be either a binary operation, for instance, $a - b$, or a sign, for instance, $-b$.

Practical Rule   ■   **$+$ and $-$**

$+$ or $-$ are binary operations when preceded and followed by a symbol or an empty group, that is, `{ }`.

So, for instance, in

$$(A + BC)x + \qquad Cy = 0,$$
$$Ex + (F + G)y = 23.$$

which is typed as (see the `alignat*` environment in Section 5.3.1)

```
\begin{alignat*}{2}
   (A + B C)x &{}+{} &C       &y = 0,\\
           Ex &{}+{} &(F + G)&y = 23.
\end{alignat*}
```

we use the empty groups, { }, to tell LaTeX that the second + in line 1 and the first + in line 2 of the formula are binary operations. If we leave out the empty groups, and type instead

```
\begin{alignat*}{2}
   (A + B C)x &+ &C       &y = 0,\\
           Ex &+ &(F + G)&y = 23.
\end{alignat*}
```

we get

$$(A + BC)x+ \qquad Cy = 0,$$
$$Ex+(F + G)y = 23.$$

This problem often arises in split formulas, for example if the formula is split just before a $+$ or $-$, you should start the next line with {}+ or {}-.

The $\mid$ symbol can play several different roles in a formula, so LaTeX provides separate commands to specify the symbol's meaning.

Practical Rule ■ The four roles of the | symbol

- | ordinary math symbol

- \mid   binary relation

- \left| left delimiter (also \lvert)

- \right| right delimiter (also \rvert)

Note the differences between the spacing in $a|b$, typed as `$a  | b$`, and in $a \mid b$, typed as `$a \mid b$`.

4.8.3   Spacing commands

There are some situations where LaTeX cannot typeset a formula properly and you have to add spacing commands. LaTeX provides a variety of spacing commands, listed in Section B.9. The negative spacing commands remove space by "reversing the print head".

The \quad and \qquad commands are often used to adjust aligned formulas or to add space before text in a formula. The size of \quad (= 1 em) and \qquad (= 2 em) depends on the current font.

The \, and \! commands are the most useful for fine tuning formulas.

Example 1   In Section 1.7, we type the formula $\int_0^\pi \sin x \, dx = 2$ as

`$\int_{0}^{\pi} \sin x \, dx = 2$`

Notice the thinspace spacing command \, between \sin x and dx. Without the command, LaTeX would have crowded $\sin x$ and dx: $\int_0^\pi \sin x dx = 2$.

Example 2   In $\sqrt{5}$side, typed as `$\sqrt{5} \text{side}$`, the square root $\sqrt{5}$ is too close to side. So type it as `$\sqrt{5} \, \text{side}$`, which typesets as $\sqrt{5}\,$ side.

Example 3   In $f(1/\sqrt{n})$, typed as `$f(1 / \sqrt{n})$`, the square root almost touches the closing parenthesis. To correct it, type `$f(1 / \sqrt{n}\,)$`, which typesets as $f(1/\sqrt{n}\,)$.

Example 4   In $\sin x/\log n$, the division symbol / is too far from $\log n$, so type

`$\sin x / \! \log n$`

which prints $\sin x/\log n$.

There is one more symbol that may need special spacing, the colon, provided by the \colon command; it is used in formulas such as `$f \colon A \to B$`; this typesets as $f\colon A \to B$. Observe that `$f: A \to B$` typesets as $f : A \to B$. The spacing is awful.

4.9   Building new symbols

4.9.1   Stacking symbols

To place any symbol above, or below, any other, for instance, $\overset{u}{\sim}$, use the \overset command. It takes two arguments—the first argument is set in a smaller size above the second argument. The spacing rules of the symbol in the second argument remain valid, i.e., the type remains the same. Since $\sim$ is a binary relation, so is $\overset{u}{\sim}$. The \underset command is the same except that the first argument is set under the second argument. For example,

$$\overset{\alpha}{a} \qquad \underset{.}{X}$$

are typed as

```
\[ \overset{\alpha}{a} \qquad \underset{\boldsymbol{\cdot}}{X} \]
```

For the \boldsymbol command, see Section 4.10.

You can also use these commands with binary relations, as in

$$f(x) \overset{\text{def}}{=} x^2 - 1$$

which is typed as

```
\[ f(x) \overset{ \text{def} }{=} x^{2} - 1 \]
```

Since = is a binary relation, $\overset{\text{def}}{=}$ becomes a binary relation, as shown by the spacing on either side. (\text{def} will follow the surrounding text style, so if it's in a theorem, it will be italic; better to use \textrm.)

Avoid these inline, see Rule 8 for formulas in Section 1.13!

4.9.2   Negating and side-setting symbols

You can *negate* with the \not command; for instance, $a \notin b$ and $a \neq b$ are typed as $a \not\in b$ and $a \not= b$, respectively. It is preferable, however, to use the negated symbols $\notin$, typed as $\notin$, and $\neq$, typed as $\ne$. See the negated binary relations table in Section B.2. For instance, "a does not divide b", $a \nmid b$, should be typed as $a \nmid b$, not as $a \not\mid b$, which typesets as $a \,|b$.

LaTeX provides the \sideset command to set symbols at the corners of large operators other than the "corners" (the last four delimiters in Section B.6). This command takes three arguments:

```
\sideset{ _{ll}^{ul} }{ _{lr}^{ur}} }{large_op}
```

where ll stands for the symbol to be placed at the lower left, ul for upper left, lr for lower right, and ur for upper right; large_op is a large operator. These two examples,

$$\prod_a^c \text{ and } {}^e\prod$$

are typed as

```
\[
   \sideset{}{_{a}^{c}}{\prod}\text{ and }\sideset{^{e}}{}{\prod}
\]
```

Note that the first two arguments are compulsory, although one or the other can be empty, while the third argument must contain the large operator.

Here is a more meaningful example:

`$\sideset{}{'}{\sum}_{\substack{ i < 10\\ j < 10 } } x_{i}z_{j}$`

is typeset as $\sideset{}{'}{\sum}_{\substack{i<10\\j<10}} x_i z_j$

Thus, \sideset helps in mixing sub- and superscripts in "limit" positions with others in "nolimit" positions, allowing for a total of six positions. Try

$$\sideset{_{a}^{c}}{_{e}^{i}}{\prod}_{n}^{r}$$

typed as `$\sideset{_{a}^{c}}{_{e}^{i}}{\prod}_{n}^{r}.`

Recall Rule 8 for formulas in Section 1.13!

4.10   Math alphabets and symbols

In Section 4.8, we discussed the classification of math symbols in the context of spacing. The symbols in a formula can also be classified as *characters from math alphabets* and *math symbols.* In the formula $A = \{x \in X \mid x\beta \geq xy > (x+1)^2 - \alpha\}$ the following characters come from math alphabets:

$$A \quad x \quad X \quad y \quad 1 \quad 2$$

whereas these characters are math symbols:

$$= \quad \{ \quad \in \quad \mid \quad \beta \quad \geq \quad > \quad (\quad + \quad) \quad - \quad \alpha \quad \}$$

4.10.1   Math alphabets

The letters and digits typed in a formula come from *math alphabets.* LaTeX's default math alphabet is Computer Modern math italic for *letters.* In the formula $x^2 \vee y_3 = \alpha$, the characters x and y come from this math alphabet. The default math alphabet for *digits* is Computer Modern roman and the digits 2 and 3 in this formula are typeset in Computer Modern roman.

LaTeX has a number of commands to switch type style in math. The two most important commands select the bold and italic versions:

Command	Math alphabet	Example
\mathbf{\gamma\ and\ \Gamma}}	math bold	$\boldsymbol{\gamma}$ **and** $\boldsymbol{\Gamma}$
\mathit{\gamma\ and\ \Gamma}	math italic	γ *and* $\mathit{\Gamma}$

These commands change the style of letters, numbers, and upper case Greek characters.

Math roman is used in formulas for operator names, such as $\sin$ in $\sin x$, and for text. For operator names, you should use the \DeclareMathOperator command or the *-ed version, which set the name of the operator in math roman, and also provide the proper spacing; see Section 7.1.8. For text, you should use the \text command; see Sections 1.7 and 4.3.3.

4.10.2   Math symbol alphabets

You may have noticed that α was not classified as belonging to an alphabet in the example at the beginning of this section. Indeed, α is treated by LaTeX as a math symbol rather than as a member of a math alphabet. There is a bold version, but you must use the `\boldsymbol` command to produce it. For instance, $\boldsymbol{\alpha}_{\boldsymbol{\beta}}$, is typed as

`$\boldsymbol{\alpha}_{\boldsymbol{\beta}}$`

Note that β appears in a small size in $\boldsymbol{\alpha}_{\boldsymbol{\beta}}$.

Four "alphabets of symbols" are built into LaTeX.

Greek The examples α and Γ are typed as `$\alpha$` and `$\Gamma$`. See Section B.1 for the symbol table.

Calligraphic An uppercase-only alphabet invoked with the `\mathcal` command. The examples $\mathcal{A}$ and $\mathcal{E}$ are typed as `$\mathcal{A}$` and `$\mathcal{E}$`.

Euler Fraktur Invoked by the `\mathfrak` command. The examples $\mathfrak{p}$ and $\mathfrak{P}$ are typed as `$\mathfrak{p}$` and `$\mathfrak{P}$`.

Blackboard bold Uppercase-only math alphabet, invoked with `\mathbb`. The examples $\mathbb{A}$ and $\mathbb{C}$ are typed as `$\mathbb{A}$` and `$\mathbb{C}$`.

5

Displayed formulas

We have discussed inline formulas in the previous chapter. An *equation* is the simplest form of a formula that is not inline; see Section 1.8.1.

LaTeX is about typesetting formulas. It knows a lot about typesetting inline formulas, but not much about how to display a multiline formula to best reflect its meaning in a visually pleasing way. So you have to decide the visual structure of a multiline formula and then use the tools provided by LaTeX to code and typeset it.

For many documents the three constructs of Section 1.8.1 suffice: *simple* and *annotated alignments*, and the *cases* construct.

5.1 Columns

Multiline formulas are displayed in *columns*. The columns are either *adjusted*, that is, centered, or set flush left or right, or *aligned*, that is, an alignment point is designated for each column and for each line. Moreover, the columns are either separated by the *intercolumn space* or adjacent with no separation.

5.1.1   One column

As in Section 1.8.3, we start with the simple align:

$$r^2 = s^2 + t^2,$$
$$2u + 1 = v + w^{\alpha}.$$

This is a single column, aligned at the $=$ signs, and coded with the `align` environment.

Another important single column environment is `gather`. It groups a number of one-line formulas, each centered on a separate line:

(1) $$x_1 x_2 + x_1^2 x_2^2 + x_3,$$

(2) $$x_1 x_3 + x_1^2 x_3^2 + x_2,$$

(3) $$x_1 x_2 x_3.$$

Formulas (1)–(3) are typed as follows:

```
\begin{gather}
 x_{1} x_{2}+x_{1}^{2} x_{2}^{2} + x_{3},\label{E:1.1}\\
 x_{1} x_{3}+x_{1}^{2} x_{3}^{2} + x_{2},\label{E:1.2}\\
 x_{1} x_{2} x_{3}.\label{E:1.3}
\end{gather}
```

Practical Rule ■ `gather` environment

1. Lines are separated with \\. Do not type a \\ at the end of the last line!

2. Each line is numbered unless it has a \tag or \notag on the line before the line separator \\.

3. No blank lines are permitted within the environment.

The `gather*` environment is like `gather`, except that all lines are unnumbered. The command \tag may still be used.

5.1.2   Two columns

The annotated align, also in Section 1.8.3, is also coded with the `align` environment,

$$
\begin{aligned}
x &= x \wedge (y \vee z) && \text{(by distributivity)} \\
&= (x \wedge y) \vee (x \wedge z) && \text{(by condition (M))} \\
&= y \vee z
\end{aligned}
$$

has two columns. The first column is aligned like our example of simple align, but the second column is aligned flush left. There is a sizeable intercolumn space.

5.1.3   Adjusted columns

An *adjusted column* is either set *centered*, or *flush left*, or *flush right*. This can happen by default, built into the environment, or so specified in the code.

In the displayed formula

$$x_1 x_2 + x_1^2 x_2^2 + x_3,$$
$$x_1 x_3 + x_1^2 x_3^2 + x_2$$

typeset with the gather environment, by default, all lines are centered. This is coded as

```
\begin{gather*}
   x_{1} x_{2} + x_{1}^{2} x_{2}^{2} + x_{3},\\
   x_{1} x_{3} + x_{1}^{2} x_{3}^{2} + x_{2}
\end{gather*}
```

On the other hand, in

$$\begin{pmatrix} 1 & 100 & 115 \\ 201 & 0 & 1 \end{pmatrix}$$

coded with the array environment:

```
\begin{equation*}
   \left(
      \begin{array}{lcr}
         1 & 100 & 115 \\
         201 & 0 & 1
      \end{array}
   \right)
\end{equation*}
```

the first column is flush left, the second centered, the third flush right; see Section 5.3.4.

5.1.4   Aligned columns

Aligned columns, on the other hand, are only of one kind, aligned by you. For instance,

$$f(x) = x + yz \qquad\qquad g(x) = x + y + z$$
$$h(x) = xy + xz + yz \qquad k(x) = (x + y)(x + z)(y + z)$$

is coded with the alignat environment. It has two aligned columns, both aligned at the = sign.

5.2   Some general rules

5.2.1   General rules

Even though you have only seen a few examples of multiline math environments, I would like to point out now that the multiline math environments and subsidiary math environments share a number of rules.

Practical Rule ■ **Multiline math environments**

1. Lines are separated with \\. Do not type a \\ at the end of the last line!

2. No blank lines are permitted within an environment.

3. If an environment contains more than one formula, then, as a rule, each formula is numbered separately. If you add a \label command to a line, then the equation number generated for that line can be cross-referenced.

4. You can suppress the numbering of a line by using a \notag command on the line.

5. You can also override numbering with the \tag command, which works just as it does for equations; see Section 1.8.2.

6. \tag and \label should always precede the line separator \\ for lines that are regarded as formulas in their own right. For instance, the lines of the multline environment cannot be individually numbered or tagged. The \tag command works for individual lines, not for the environment as a whole.

7. For cross-referencing, use \label, \ref, and \eqref in the same way you would for an equation (see Section 1.8.2).

8. Each multiline math environment has a *-ed form, which suppresses numbering. Individual formulas can still be \tag-ged.

5.2.2   Breaking and aligning formulas

You do not have to know where and how to break inline formulas because LaTeX does all the work for you.

Unfortunately, multiline formulas are different. LaTeX gives you excellent tools for displaying multiline formulas, but offers you no advice on deciding where to break a long formula into lines. You, the author, are the judge of where to break a long formula so that the result is mathematically informative and follows the traditions of mathematical typesetting.

Practical Rule ■ **Breaking displayed formulas**

1. Try to break a long formula *before* a binary relation or binary operation.

2. If you break a formula before a + or -, start the next line with {}+ or {}-.

3. If you break a formula within a bracket, indent the next line so that it begins *to the right of* the opening bracket.

Here is an illustration of the third rule:

$$f(x, y, z, u) = [(x + y + z) \times (x^2 + y^2 + z^2 - 1)$$
$$\times (x^3 + y^3 + z^3 - u) \times (x^4 + y^4 + z^4 + u)]^2$$

The rules for aligning columns are similar.

Practical Rule ■ **Aligning columns**

1. Try to align columns at a binary relation or a binary operation.

2. If you align a column at a binary relation, put the & symbol immediately *to the left* of the binary relation.

3. If you align a column at the binary operation + or -, put the & symbol to the left of the binary operation.

5.3 Aligned columns

The lines of multiline formulas are naturally divided into columns. In this section, we discuss how to typeset such formulas with *aligned columns.* All of these constructs are implemented with the align math environment and its variants.

In Section 1.8.3, you saw two simple, one-column examples of aligned columns—which we called *simple alignment*—and a special case of aligned columns—which we called *annotated alignment*.

The align environment can also create multiple aligned columns. The number of columns is restricted only by the width of the page. In the following example, there are two aligned columns:

$$\text{(4)} \qquad \begin{aligned} f(x) &= x + yz & g(x) &= x + y + z \\ h(x) &= xy + xz + yz & k(x) &= (x + y)(x + z)(y + z) \end{aligned}$$

typed as

```
\begin{align}\label{E:mm3}
   f(x) &= x + yz        & g(x) &= x + y + z\\
   h(x) &= xy + xz + yz  & k(x) &= (x + y)(x + z)(y + z)
      \notag
\end{align}
```

Use Figure 5.1 to visualize how the alignment points in the source turn into alignment points in the typeset formula and the role played by the intercolumn space. Remember that the visual layout of the source is for your benefit only.

In a multicolumn align environment, the ampersand (&) plays two roles. It is a mark for the *alignment point* and it is also a *column separator*. In the line

```
   f(x) &= x + yz        & g(x) &= x + y + z
```

the two columns are

```
f(x) &= x + yz
```

and

```
                        g(x) &= x + y + z
```

In each column, we use a single ampersand to mark the alignment point.
Of the three & symbols in the previous example,

- The first & marks the *alignment point* of the first column.

- The second & is a *column separator* that separates the first and second columns.

- The third & marks the *alignment point* of the second column.

I use the convention of typing a space on the left of an alignment point & and no space
on the right, and of putting spaces on both sides of & as a column separator.

    If the number of columns is three, then there should be five &'s in each line. Even-
numbered &'s are column separators and odd-numbered &'s are alignment points.

_____

Practical Rule ■ **Ampersands**

If there are n aligned columns, then each line should have at most $2n - 1$ amper-
sands. Even-numbered &'s are column separators; odd-numbered &'s mark the align-
ment points.

_____
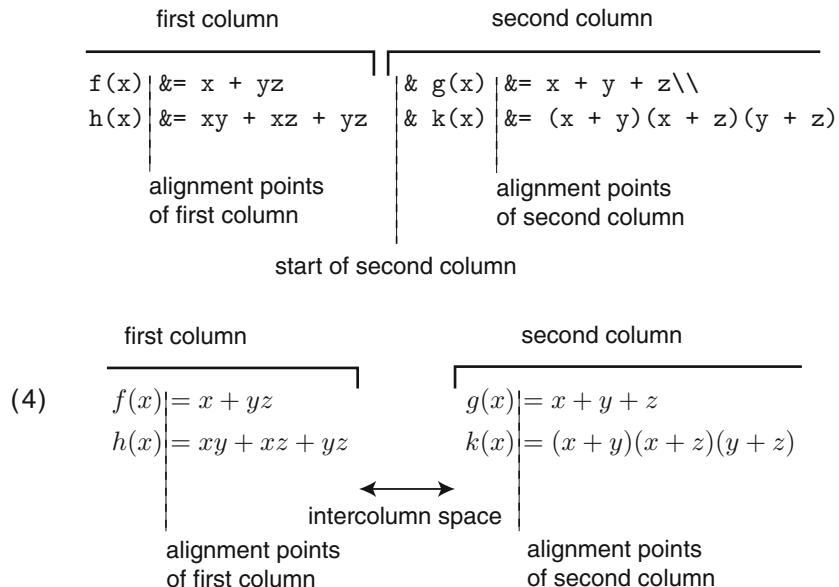
Figure 5.1: Two aligned columns: source and typeset.

So for a single aligned column, you have to place one alignment point for each line. For two aligned columns, you have to place at most three alignment points for each line. The beginning of the line to the second & is the first column, then from the second & to the end of the line is the second column. Each line of each column has an alignment point marked by &.

5.3.1 The `alignat` *environment*

Another variant of the `align` environment is the `alignat` environment, which is one of the most important alignment environments. While the `align` environment calculates how much space to put between the columns, the `alignat` environment leaves spacing up to the user. It is important to note that the `alignat` environment has a required argument, the number of columns.

Here is formula (4) typed with the `alignat` environment:

```
\begin{alignat}{2}\label{E:mm3A}
   f(x) &= x + yz       & g(x) &= x + y + z\\
   h(x) &= xy + xz + yz & k(x) &= (x + y)(x + z)(y + z)
   \notag
\end{alignat}
```

which typesets as

$$
\begin{aligned}
f(x) &= x + yz \qquad\qquad g(x) = x + y + z \\
h(x) &= xy + xz + yzk(x) = (x + y)(x + z)(y + z)
\end{aligned}
\tag{5}
$$

This attempt did not work very well because `alignat` did not separate the two formulas in the second line. So you must provide the intercolumn spacing. For instance, if you want a \qquad space between the columns, as in

$$
\begin{aligned}
f(x) &= x + yz \qquad\qquad g(x) = x + y + z \\
h(x) &= xy + xz + yz \qquad k(x) = (x + y)(x + z)(y + z)
\end{aligned}
\tag{6}
$$

then type the formula as

```
\begin{alignat}{2}\label{E:mm3B}
   f(x) &= x + yz                  & g(x) &= x + y + z\\
   h(x) &= xy + xz + yz \qquad & k(x) &= (x+y)(x+z)(y+z)
   \notag
\end{alignat}
```

The `alignat` environment is especially appropriate when annotating formulas where you would normally want a \quad between the formula and the text. To obtain

$$
\begin{aligned}
x &= x \wedge (y \vee z) \qquad && \text{(by distributivity)} \\
&= (x \wedge y) \vee (x \wedge z) \quad && \text{(by condition (M))} \\
&= y \vee z
\end{aligned}
\tag{7}
$$

type
```
\begin{alignat}{2}\label{E:mm4}
   x &= x \wedge (y \vee z) &
   &\quad\text{(by distributivity)}\\
     &= (x \wedge y) \vee (x \wedge z) & &
       \quad\text{(by condition (M))}\notag\\
     &= y \vee z \notag
\end{alignat}
```

alignat is very important for typing systems of equations such as

$$(8) \qquad\qquad (A + BC)x + \qquad Cy = 0,$$

$$(9) \qquad\qquad Ex + (F + G)y = 23.$$

typed as follows:
```
\begin{alignat}{2}
   (A + B C)x &+{} &C       &y = 0,\\
            Ex &+{} &(F + G)&y = 23.
\end{alignat}
```

As a last example, consider

$$(10) \qquad\qquad a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \qquad\qquad = y_1,$$

$$(11) \qquad\qquad a_{21}x_1 + a_{22}x_2 \qquad + a_{24}x_4 = y_2,$$

$$(12) \qquad\qquad a_{31}x_1 \qquad + a_{33}x_3 + a_{34}x_4 = y_3.$$

typed as
```
\begin{alignat}{4}
  a_{11}x_1 &+ a_{12}x_2 &&+ a_{13}x_3 &&
    &&= y_1,\\
  a_{21}x_1 &+ a_{22}x_2 &&              &&+ a_{24}x_4
    &&= y_2,\\
  a_{31}x_1 &              &&+ a_{33}x_3 &&+ a_{34}x_4
    &&= y_3.
\end{alignat}
```

Note that the argument of alignat does not have to be precise. If you want two columns, the argument can be 2, or 3, or any larger number. If you want to, you can simply type 10 and just ignore the argument. You can define a new environment (see Section 7.2.1) that does just that.

5.3.2   *Inserting text*

The \intertext command places one or more lines of text in the middle of an aligned environment.

For instance, to obtain

$$(13) \qquad h(x) = \int \left(\frac{f(x) + g(x)}{1 + f^2(x)} + \frac{1 + f(x)g(x)}{\sqrt{1 - \sin x}} \right) dx$$

The reader may find the following form easier to read:

$$= \int \frac{1 + f(x)}{1 + g(x)} \, dx - 2 \arctan(x - 2)$$

you would type

```
\begin{align}\label{E:mm5}
     h(x) &= \int \left(
                        \frac{ f(x) + g(x) }
                             {1 + f^{2}(x)} +
                        \frac{1 + f(x)g(x)}
                             { \sqrt{1 - \sin x} }
                 \right) \, dx\\
   \intertext{The reader may find the following form
        easier to read:}
     &= \int \frac{1 + f(x)}
                  {1 + g(x)}
          \, dx - 2 \arctan(x - 2) \notag
\end{align}
```

Notice how the equal sign in the first formula is aligned with the equal sign in the second formula even though a line of text separates the two.

Here is another example, this one using `align*`:

$$f(x) = x + yz \qquad\qquad\qquad g(x) = x + y + z$$

The reader may also find the following polynomials useful:

$$h(x) = xy + xz + yz \qquad\qquad k(x) = (x + y)(x + z)(y + z)$$

is typed as

```
\begin{align*}
   f(x) &= x + yz & \qquad g(x) &= x + y + z\\
   \intertext{The reader may also find the following
    polynomials useful:}
   h(x) &= xy + xz + yz
                   & \qquad k(x) &= (x + y)(x + z)(y + z)
\end{align*}
```

The `\intertext` command must follow a line separator command, `\\` or `\\*`. If you violate this rule, you get the error message

```
! Misplaced \noalign. \intertext #1->\noalign
            {\penalty \postdisplaypenalty \vskip ...
```

5.3.3   Matrices

Use the matrix environment to typeset matrices. For example,

```
\begin{equation*}
   \left(
   \begin{matrix}
      a + b + c & uv    & x - y & 27\\
      a + b     & u + v & z     & 1340
   \end{matrix}
   \right) =
   \left(
   \begin{matrix}
      1   & 100 & 115 & 27\\
      201 & 0   & 1   & 1340
   \end{matrix}
   \right)
\end{equation*}
```

produces

$$\begin{pmatrix} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 1340 \end{pmatrix} = \begin{pmatrix} 1 & 100 & 115 & 27 \\ 201 & 0 & 1 & 1340 \end{pmatrix}$$

If you use matrix on its own, i.e., outside a math environment,

```
\begin{matrix}
   a + b + c & uv    & x - y & 27\\
   a + b     & u + v & z     & 134
\end{matrix}
```

you get the error message

```
! Missing $ inserted.
<inserted text>
                  $
l.5 \begin{matrix}
```

obliquely reminding you that matrix is a subsidiary math environment.

The matrix subsidiary math environment provides a matrix of up to 10 centered columns. If you need more columns, you have to ask for them. The following example sets the number of columns to 12:

```
\begin{equation}\label{E:mm12}
   \setcounter{MaxMatrixCols}{12}
   \begin{matrix}
      1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12\\
      1 & 2 & 3 & \hdotsfor{7}                & 11 & 12
   \end{matrix}
\end{equation}
```

produces

(14)
$$\begin{array}{cccccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & \multicolumn{6}{c}{\dotfill} & 11 & 12 \end{array}$$

You can have dots span any number of columns with the \hdotsfor command. The argument of the command specifies the number of columns to fill (which is one more than the number of &'s the command replaces). The \hdotsfor command must appear either at the beginning of a row or immediately following an ampersand (&). If you violate this rule, you get the error message

```
! Misplaced \omit.
\multispan #1->\omit
                    \mscount #1\relax \loop \ifnum
                                            \mscount ...
l.12 \end{equation}
```

Matrix variants

A matrix can be enclosed by delimiters (see Section 4.4.1) in a number of different ways:

$$\begin{matrix} a+b+c & uv \\ a+b & c+d \end{matrix} \qquad \begin{pmatrix} a+b+c & uv \\ a+b & c+d \end{pmatrix} \qquad \left(\begin{matrix} a+b+c & uv \\ a+b & c+d \end{matrix} \right]$$

The first matrix is typed as

```
\begin{matrix}
   a + b + c & uv\\
   a + b     & c + d
\end{matrix}
```

The second uses pmatrix, and the third is typed as

```
\left(
   \begin{matrix}
      a + b + c & uv\\
      a + b & c + d
   \end{matrix}
\right]
```

Small matrix

A matrix inline is too large. Instead, use the smallmatrix environment. Compare $\left(\begin{smallmatrix} a+b+c & uv \\ a+b & c+d \end{smallmatrix} \right)$, typed as

```
$\begin{pmatrix}
   a + b + c & uv\\
   a + b     & c + d
\end{pmatrix}$
```

with the small matrix $\left(\begin{smallmatrix} a+b+c & uv \\ a+b & c+d \end{smallmatrix}\right)$, typed as

```
$\left(
\begin{smallmatrix}
   a + b + c & uv\\
   a + b     & c + d
\end{smallmatrix}
\right)$
```

5.3.4   *Arrays*

The `array` environment is similar to the `matrix` subsidiary math environment. However, for an `array` you must specify the alignment of each column and you have more options to customize it.

The first matrix in Section 5.3.3 would be typed as follows using the `array` subsidiary math environment:

```
\begin{equation*}
   \left(
   \begin{array}{cccc}
      a + b + c & uv    & x - y & 27\\
      a + b     & u + v & z     & 134
   \end{array}
   \right)
\end{equation*}
```

Practical Rule ■ `array` **subsidiary math environment**

1. Adjacent columns are separated by an ampersand (`&`).

2. The argument is a series of the letters `l`, `r`, or `c`, signifying that the corresponding column in the array should be set flush left, flush right, or centered, respectively.

CHAPTER

6

Documents

In this chapter, we take up the organization of a document.

Section 6.1 discusses document structure in general, Section 6.2 presents the preamble. Section 6.3 discusses the top matter, in particular, the abstract environment. Section 6.4 presents the main matter, including sectioning, cross-referencing, tables, and figures. Section 6.5 covers the back matter, including the bibliography. In Section 6.5.4 we discuss the Table of Contents, lists of tables and figures. Finally, in Section 6.6, we discuss in detail the amsart document class.

6.1 The structure of a document

As we introduced it in Section 1.9, the source file of a LaTeX document is divided into two main parts: the preamble and the body; see Figure 1.1. The body is divided into three parts: the top matter, the main matter, and the back matter.

6.2   The preamble

The preamble contains the crucial \documentclass line, specifying the document class and the options that modify its behavior. For instance,

```
\documentclass[reqno]{amsart}
```

loads the document class amsart with the reqno option, which places the equation numbers on the right.

The \documentclass command is usually followed by the \usepackage commands, which load LaTeX enhancements called *packages*. For instance,

```
\usepackage{graphicx}
```

loads the graphicx package; see Section 6.4.2.
\usepackage commands can be combined:

```
\usepackage{amssymb,latexsym}
```

Document class files have a cls extension, whereas package files are designated by the sty extension. The document class amsart is defined in the amsart.cls file, the graphicx package is defined in the graphicx.sty file.

The preamble normally contains any user-defined commands and the proclamation definitions; see Sections 1.12, 3.3 and Chapter 7. Some commands can only be in the preamble. \DeclareMathOperator is such a command; see Section 7.1.8. If you place it in the body, you get an error message:

```
! LaTeX Error: Can be used only in preamble.
l.103 \DeclareMathOperator
```

6.3   Top matter

The top matter of a document is part of the document body and, as a rule, it contains the material used to create the "title page" and, optionally, an abstract.

We discussed briefly the top matter in Section 1.9, and we continue discussing it in much more detail in Section 6.6.1.

Place the abstract environment *before* the \maketitle command; see Figure 1.1. If you forget to place it there, you get the warning

```
Class amsart Warning:
              Abstract should precede \maketitle in AMS
documentclasses; reported on input line 21.
```

and the abstract is typeset wherever the abstract environment happens to be placed.

If the abstract and the "footnotes" from the top matter fill the first page, the second page has no running head. To fix this, follow the \maketitle command with the \clearpage command (see Section 2.6.3).

6.4   Main matter

The main matter contains most of the essential parts of the document, including the appendices.

We discuss now how to structure the main matter. We describe sectioning in Section 6.4.1 and tables and figures in Section 6.4.2.

6.4.1   Sectioning

The main matter of a document is divided into *sections*.

Sections

LaTeX is instructed to start a section with the \section command, which takes the title of the section as its argument. This argument is used for Table of Contents, which means that you need to protect fragile commands with the \protect command; see Sections 2.2.3 and 6.5.4. LaTeX automatically assigns a section number and typesets the section number followed by the section title.

Other sectioning commands

A section can be subdivided into *subsections,* which can themselves be divided into *subsubsections, paragraphs,* and *subparagraphs.* Subsections are numbered within a section (in Section 1, they are numbered 1.1, 1.2, and so on).

If the first sectioning command in your document is \subsection, the subsections are numbered 0.1, 0.2, and so on. If in the first section of your document the first sectioning command is \subsubsection, the subsubsections are numbered 1.0.1, 1.0.2, and so on. Both are clearly undesirable.

Any sectioning command can be followed by a \label command so that you can refer to the number (if any) generated by LaTeX and the page on which it appears; see Section 1.8.2.

The form of sectioning commands

All sectioning commands take one of the following two forms, illustrated by the \section command:

The usual form is \section{title} where title is the section title, of course. You need to protect any fragile commands in title with the \protect command; see Section 2.2.3.

If the *-ed version: \section*{title} is used, there are no section numbers printed. Remember that if you * a section, all subsections, and so on, must also be *-ed within it to avoid having strange section numbers.

Appendix

The \appendix command, in the main matter, marks the beginning of the appendices, if any. After the \appendix command, the \section command starts an appendix:

```
\appendix
\section{A proof of the Main Theorem}\label{A:geom}
```

This produces Appendix A with the given title, typeset just like a section.

Note that appendices can be labeled and cross-referenced like any other section. In an appendix, subsections are numbered A.1, A.2, and so on, subsubsections within A.1 are numbered A.1.1, A.1.2, and so on.

6.4.2   Floating tables and illustrations

Many documents contain tables and illustrations. These must be treated in a special way since they cannot be broken across pages. If necessary, LATEX moves—floats— a table or an illustration to the top or bottom of the current or the next page if possible and further away if not.

LATEX provides the table and the figure environments for typesetting floats. The two are identical except that the figure environments are captioned Figure 1, Figure 2, and so on, whereas the table environments are numbered as Table 1, Table 2, and so on.

Tables

A table environment is set up as follows:

```
\begin{table}
   \caption{title}\label{Ta:xxx}
   Place the table here
\end{table}
```

The \caption command is optional and should precede the table. The optional \label command must follow the command \caption. The label is used to reference the table's number. A table environment can have more than one table, each with its own caption.

The table environment is primarily used for tables made with the tabular environment; see Section 3.5. There are many examples of tables in this book.

Figures

Illustrations, also called *graphics* or *figures*, include drawings, scanned images, digitized photos, and so on. These can be inserted with a figure environment. Using the graphicx package, a typical figure is specified as follows:

```
\begin{figure}
   \centering\includegraphics{file}
   \caption{title }\label{Fi:xxx}
\end{figure}
```

The illustration `circle.pdf` is included with the command

```
\includegraphics{circle}
```

without the extension.

If you have to scale `circle.pdf`, say to 68% of its original size, use the command

```
\includegraphics[scale=.68]{circle}
```

Float control

The `table` and `figure` environments can have an optional argument, with which you can influence LaTeX's placement of the typeset table. The optional argument consists of one to four letters: `b` (the bottom of the page), `h` (here, where the environment appears in the text), `t` (the top of the page), `p` (a separate page). For instance,

```
\begin{table}[ht]
```

requests LaTeX to place the table "here" or at the "top" of a page. The default is `[tbp]` and the order of the optional arguments is immaterial. If `h` is specified, it takes precedence, followed by `t` and `b`. `[!h]` requests that this table or figure be placed where it is in the source file. Similarly, you can use `[!t]` or `[!b]`.

Your demands and LaTeX's float mechanism can conflict with one another with the result that LaTeX does not place material where you want it. The default values of the float placement parameters are good only for documents with a small number of floats.

Practical Tip 36. Combining two tables or illustrations into one maybe helpful.

6.5   Back matter

The back matter of a document is very simple, as a rule. It is either empty or consists of only a bibliography. In this section, we discuss the *bibliography* and a very simple *index*.

6.5.1   Bibliographies

The simplest way to typeset a bibliography is to type it directly into the document. You type the bibliographic items in a `thebibliography` environment, as in the following examples.

```
\begin{thebibliography}{9}
\bibitem{gC13}%G. Cz\'edli~\cite{gC13}
   G. Cz\'edli, \emph{Patch extensions of slim lattices},
   Algebra Universalis \textbf{88} (2013), 255--280.
\bibitem{sF98}%Soo-Key Foo~\ref{sF98}
   Soo-Key Foo, \emph{Lattice constructions},
   Ph.D. thesis, University of Winnebago, 1998.
\bibitem{sF05}%Soo-Key Foo~\ref{sF05}
   \bysame, \emph{Composition of functions}.
   Proc. Conf. on Universal Algebra (Kingston, 2005).
\bibitem{gM12}%George~A. Menuhin~\ref{gM12}
   George~A. Menuhin, \emph{Universal algebra}.
   University Series in Higher Mathematics, vol.~158,
   D.~Van Nostrand, Princeton, 2012.
\end{thebibliography}
```

Figure 6.1 shows a typeset version of this bibliography.

The `thebibliography` environment takes an argument—in the previous example, this argument is 9—telling LaTeX that the widest reference number it must generate is one digit wide. For more than 9 but fewer than 100 items, use 99, and so on.

If the argument of `\begin{thebibliography}` is missing, you get the error message

```
! LaTeX Error: Something's wrong--perhaps a missing \item.
```

Each bibliographic item is introduced with `\bibitem`, which is used just as the `\label` command. In your text, use `\cite`, just as `\eqref`—it provides the number enclosed in brackets. So if the 4th bibliographic item is introduced with `\bibitem{gM12}` then `\cite{gM12}` refers to that item and typesets it as [4]. The bibliography of the document itself is automatically numbered by LaTeX. It is up to the author to make sure that the listing of the bibliographic items is in the proper order.

Practical Tip 37. Do not leave a space in a `\cite` command: `\cite{gm12␣}` produces [?] indicating an unknown reference.

REFERENCES

[1] G. Czédli, *Patch extensions and trajectory colorings of slim rectangular lattices*, Algebra Universalis **88** (2013), 255–280.
[2] Soo-Key Foo, *Lattice constructions*, Ph.D. thesis, University of Winnebago, 1998.
[3] _____, *Composition of functions*. Proc. Conf. on Universal Algebra (Kingston, 2005).
[4] George A. Menuhin, *Universal algebra*. University Series in Higher Mathematics, vol. 158, D. Van Nostrand, Princeton, 2012.

Figure 6.1: Some bibliographic entries

 Practical Tip 38. I use the convention that the label for a \bibitem consists of the two initials of the author and the year of publication. The first cited publication by George A. Menuhin in 2012 has the label gM12 and the second would be gM12a. Of course, you can use any label you choose, but such conventions make the items easier to use and reuse.

You can use \cite to cite two or more items in the form

\cite{gC13,sF05}

which typesets as [1, 3]. There is also an optional argument for \cite to specify additional information. For example,

\cite[pages~2--15]{sF05}

typesets as [3, pages 2–15].

   If you wish to use labels rather than numbers to identify bibliographic items, then you can specify those labels with an optional argument of the \bibitem command: \bibitem[gM12]{gM12}. If this optional argument of \bibitem is used, then the \cite command produces [gM12]. The argument of \begin{thebibliography} must be set wide enough to allow for such labels.

   If an author appears repeatedly, use the \bysame command, which replaces the author's name with a long dash followed by a thin space. For an example, see page 94.

   To rename the bibliography, see Section 7.1.6.

6.5.2 BIBT_EX

The BIBTEX application assists in compiling bibliographies for your documents extracted from bibliographic databases that can be reused and shared.

   It takes some effort to learn how to build BIBTEX bibliographic databases. See Chapter 16 of MiL4 for an introduction. However, if you have access to such databases, it is quite easy to use them.

The database

A BIBTEX database is a text file, such as firstdocumentb.bib in the samples folder, containing bibliographic entries. This file has only two entries:

```
@ARTICLE(gC13,
   author = "G. Czedli",
   title = "Patch extensions of slim lattices",
   journal = "Algebra Universalis",
   pages = "255--280",
   volume = 88,
   year = 2013,
        )
```

```
@ARTICLE(gG14,
   author = "G. Gratzer",
   title = "Congruences of fork extensions of lattices",
   journal = "Acta Sci. Math. (Szeged)",
   pages = "417--434",
   volume = 57,
   year = 2014,
          )
```

We dropped the accents to simplify the discussion.

Getting started

To list database entries in the bibliography, use the \cite command, see Section 6.5.1.
If you want to have a reference listed in the bibliography without a citation in the text,
then use the \nocite command. For example, \cite{gC13} includes the reference in
the bibliography and cites the entry with label gC13, whereas \nocite{gC13} includes
the reference in the bibliography but does not cite the entry. In either case, one of the
bib files specified in the argument of the \bibliography command must contain an
entry with the label gC13.

   Your document must specify the bibliography style and must name the bib files
to be used. For instance, the firstdocumentb.tex sample article contains the lines:

```
\bibliographystyle{amsplain}
\bibliography{firstdocumentb}
```

The \bibliographystyle command specifies amsplain.bst as the style and the
\bibliography command specifies the database file firstdocumentb.bib, in the
samples folder. To use several database files, separate them with commas, as in

```
\bibliography{abbrev,gg,lattice,firstdocument}
```

It is important to make sure that the bst file, the bib file(s), and the LaTeX document(s)
are in folders where BibTeX can find them. If you are just starting out, you can simply
copy all of them into one folder.

Three steps of BibTeXing

The following steps produce a typeset bibliography in your LaTeX document. We use
the firstdocumentb.tex sample article as an example.

Step 1 Typeset firstdocumentb.tex to get a fresh aux file.

Step 2 Run BibTeX on the firstdocumentb.aux file by choosing it as a menu option
in your editor in the LaTeX implementation or by clicking on its icon.

If BIBTEX cannot find

1. a crucial file, for example, the bst file,

2. a database entry corresponding to a \cite or \nocite command,

then it stops. The reason why it stopped is shown in the log window and also written to a blg (bibliography log) file, firstdocumentb.blg. Correct the error and redo the step. A successful run creates a bbl (bibliography) file, firstdocumentb.bbl.

Step 3 Typeset the LATEX document firstdocumentb.tex *twice*.

You get the same typeset document as on page .

Submitting an article

If you submit an article to a journal that specifies which BIBTEX style file to use, then you can submit the article and the BIBTEX database file, pared down of course. If this is not the case, create the bbl file with amsplain.bst and copy and paste the thebibliography environment from it into the article.

6.5.3   Simple indexes

Using the \label and \pageref commands (see Section), it is quite simple to produce a small index in a theindex environment. At each point in the text that you want to reference in the index, place a \label command. The corresponding entry in the index typesets the page number with the \pageref command.

The \item, \subitem, and \subsubitem commands create an entry, subentry, and subsubentry, respectively. If you need additional vertical spacing when the first letter changes, for instance, between the "h" entries and the "i" entries, you can use the \indexspace command. Here are some examples of index entries:

```
\begin{theindex}
\item Lakser, H., \pageref{Lakser}
\item Lattice, \pageref{Lattice_intro},
                \textbf{\pageref{Lattice}}
  \subitem distributive, \pageref{Lattice_distributive}
  \subitem modular, \pageref{Lattice_distributive},
            \textbf{\pageref{Lattice_distributive2}}
\item Linear subspace, \pageref{Linear_subspace}
\end{theindex}
```

See Figure 6.2 for the typeset index.

For a larger index, you should use the *MakeIndex* application; see Chapter 17 of MiL4.

Figure 6.2: A typeset index

6.5.4   Tables of Contents

What goes into the Table of Contents? All titles of sections, and subsections, not the short titles, whether *-ed or not.

When you typeset your document with a Table of Contents, LaTeX creates a file with the toc extension. The next time the document is typeset, the toc file is typeset too and included in your typeset document at the point where the command

`\tableofcontents`

appears in the source file, normally in the front matter. If your source file is named document.tex, the toc file is named document.toc. This file lists all the sectioning units as well as their titles and page numbers. You may have to typeset your document *three times* to get the Table of Contents right.

LaTeX adds a line to the Table of Contents, formatted like a section title, if you include the command

`\addcontentsline{toc}{section}{text_to_be_added}`

in your source file.

There are three arguments:

1. The first argument informs LaTeX that a line, the third argument, should be added to the toc file.

2. The second argument specifies how the line should be formatted in the Table of Contents. In our example, the second argument is section, so the line is formatted as a section title in the Table of Contents. The second argument must be the name of a sectioning command.

3. The third argument is the text to be added.

Fragile commands in a movable argument, such as a section title, must be protected with the `\protect` command; see Section 2.2.3.

There are also similar commands for lists of tables (`\listoftables`, lot) and figures (`\listoffigures`, lof).

6.6   *The AMS article document class*

In this section, we discuss amsart, the only document class in this book.

The next three sections discuss the top matter, split into three parts: information about the article, information about the author, and subject classification. Section 6.6.4 presents some examples.

A document class is shaped by its options. In Section 6.6.5, we discuss some of the options of amsart.

As you may recall from Section 1.9, part of the author information is moved to the end of the typeset article.

There is only one general rule.

Practical Rule ■ **Top matter commands**

You must specify the \title. All the other top matter commands are optional.

Practical Rule ■ **Top matter commands**

All top matter commands are *short*.

This means that there can be no blank lines (or \par commands) in the argument of any of these commands.

6.6.1   *The top matter: Article information*

We discuss two pieces of information about the document.

Practical Rule ■ **Title**

- Command: \title

- Separate lines with \\

- Optional argument: Short title for running head

- Do not put a period at the end of a title

- Do not put user-defined commands in the title

The typeset title is placed on the front page of the typeset document.

The *running head* is the title on odd-numbered pages, set in capital letters. If the title is more than a few words long, use an optional argument to specify a short title for the running head. Do not use \\ in the short title.

Examples:

```
\title{A construction of distributive lattices}
```

A title with a short title:

```
\title[Complete-simple distributive lattices]
{A construction of\\ complete-simple
distributive lattices}
```

Practical Rule ■ Date

- Command: \date

The typeset \date is placed on the front page of the typeset article as a footnote.
Example:

```
\date{January 22, 2014}
```

You can use the \today command to get today's date: \date{\today}. Do not use
this when you submit an article; specify the submission date.

  To suppress the date, use \date{} or omit the \date command entirely.

6.6.2   The top matter: Author information

You can include information about yourself.

Practical Rule ■ Author

- Command: \author

- Optional argument: Short form of the name for the running head

---

The typeset author is placed on the front page of the typeset document.
Examples:
An author:

```
\author{George~A. Menuhin}
```

An author with a short form of the name for the running head:

```
\author[G.\,A. Menuhin]{George~A. Menuhin}
```

We discuss shortly how to specify multiple authors.

Practical Rule ■ Address

- Command: \address
- Separate lines with \\
- Optional argument: Name of author

The typeset address is placed at the end of the typeset document.

Example:

DEPARTMENT OF APPLIED MATHEMATICS, UNIVERSITY OF WINNEBAGO, WINNEBAGO, MN 53714

which is typed as

```
\address{Department of Applied Mathematics\\
         University of Winnebago\\
         Winnebago, MN 53714}
```

Notice that the \\ line separators are replaced by commas.

If there are several authors, you can use the author's name as an optional argument of \address to avoid ambiguity.

Multiple authors

If a document has several authors, repeat the author information commands for each one. Take care that the e-mail address follows the address.

If two authors share the same address, omit the \address command for the second author, who can still have a different e-mail address and Web home page. An additional \thanks command for the first author should precede any \thanks commands (see page 103) for the second author. Since the footnotes are not marked, the argument of the \thanks command for research support should contain a reference to the author:

```
\thanks{The research of the first author was supported
        in part by NSF grant PAL-90-2466.}
\thanks{The research of the second author was supported by
        the Hungarian National Foundation for Scientific
        Research, under Grant No.~9901.}
```

Finally, if a document has too many authors to fit the running head, supply the author information for each author as usual, but explicitly specify the running heads with the \markleft command:

```
\markleft{FIRST AUTHOR ET AL.}
```

in all capitals.

If there are multiple authors, sometimes it may not be clear whose address, current address, e-mail address, or Web home page is being given. In such cases you can give the name of the author as an optional argument for these commands. For example,

Email address, Ernest T. Moynahan: `emoy@ccw.uwinnebago.edu`.

is typed as

`\email[Ernest~T. Moynahan]{emoy@ccw.uwinnebago.edu}`

Practical Rule ■ E-mail address

- Command: `\email`

- Optional argument: Name of author

The typeset e-mail address is placed at the end of the typeset document.
Example:

`\email{gmen@ccw.uwinnebago.edu}`

Practical Tip 39. Some e-mail addresses contain the special underscore character (`_`). Recall that you have to type `\_` to get `_`.

Example:

`\email{George\_Gratzer@umanitoba.ca}`

Practical Tip 40. Some older e-mail addresses contain the percent symbol (%); recall that you have to type `\%` to get %.

Example:

`\email{h1175moy\%ella@relay.eu.net}`

Practical Rule ■ Web (home) page (URL)

- Command: `\urladdr`

- Optional argument: Name of author

The typeset Web (home) page is placed at the end of the typeset document.
Example:

`\urladdr{http://www.maths.umanitoba.ca/homepages/gratzer/}`

💡 **Practical Tip 41.** Many Internet addresses contain the tilde (~), indicating the home directory of the user. Type ~ to get ~ and not \~. The symbol $\sim$ is also not acceptable.

💡 **Practical Tip 42.** If your Web (home) page address is long, go to `tinyurl.com` and make a short version.

Practical Rule ■   Research support or other acknowledgments

- Command: \thanks

- Do not specify linebreaks.

- Terminate the sentence with a period.

The typeset research support or other acknowledgments is placed on the front page of the typeset document as an unmarked footnote.

Example:

`\thanks{Supported in part by NSF grant PAL-90-2466.}`

A \thanks{} command is ignored in typesetting.

An additional \thanks command creates an unmarked footnote.

Example:

`\thanks{This is a preliminary version of this article.}`

6.6.3   The top matter: Subject information

You can supply information about the subject of the article.

The following are collected at the bottom of the first page as unmarked footnotes along with the arguments of the \thanks and \date commands.

Practical Rule ■ Subject classifications

- Command: \subjclass

- Optional argument: 2010—the default is 1991.

- amsart supplies the phrase 1991 *Mathematics Subject Classification* and a period at the end of the subject classification—with the optional argument 2010, the phrase is 2010 *Mathematics Subject Classification*

- The argument should be either a five-character code or the phrase `Primary:` followed by a five-character code, a semicolon, the phrase `Secondary:` and one or more additional five-character codes.

The typeset AMS subject classifications are placed at the bottom of the front page of the typeset article as a footnote.
Examples:

```
\subjclass[2010]{06B10}
\subjclass[2010]{Primary: 06B10; Secondary: 06D05}
```

💡 **Practical Tip 43.** The current subject classification scheme for mathematics was introduced in 2010, making the 1991 classification scheme obsolete. Thus, 2010 should be considered as a *compulsory* optional argument—maybe the only one in all of LaTeX.

💡 **Practical Tip 44.** The current subject classification scheme, MSC 2010, is available from the AMS Web site

http://www.ams.org/

> Search for MSC.

Practical Rule ■ Keywords

- Command: \keywords

- Do not indicate line breaks.

- amsart supplies the phrase *Key words and phrases.* and a period at the end of the list of keywords.

The typeset keywords are placed on the front page of the typeset article as a footnote.
Example:

```
\keywords{Complete congruence, congruence lattice}
```

6.6.4   Examples

The following examples show typical top matter commands.

Example 1   One author.
Article information:

```
\title[Complete-simple distributive lattices]
      {A construction of complete-simple\\
       distributive lattices}
\date{\today}
```

Author information:

```
\author{George~A. Menuhin}
\address{Computer Science Department\\
         University of Winnebago\\
         Winnebago, MN 53714}
\email{gmen@ccw.uwinnebago.edu}
\urladdr{http://math.uwinnebago.edu/homepages/menuhin/}
\thanks{This research was supported by
         the NSF under grant number 23466.}
```

Subject information:

```
\keywords{Complete congruence, congruence lattice}
\subjclass[2010]{Primary: 06B10; Secondary: 06D05}
```

In the `\title` command, supplying the optional argument for the running head is the rule, not the exception. Recall that the only required item is `\title`. If it is missing, you get the strange error message:

```
! Undefined control sequence.
<argument> \shorttitle


l.49 \maketitle
```

Example 2   Two authors but only the first has a Web home page. I only show the author information section here. The other commands are the same as in Example 1.
Author information:

```
\author{George~A. Menuhin}
\address{Computer Science Department\\
         University of Winnebago\\
         Winnebago, MN 53714}
\email{gmen@ccw.uwinnebago.edu}
\urladdr{http://math.uwinnebago.edu/homepages/menuhin/}
\thanks{The research of the first author was
         supported by the NSF under grant number 23466.}
\author{Ernest~T. Moynahan}
\address{Mathematical Research Institute
         of the Hungarian Academy of Sciences\\
         Budapest, P.O.B. 127, H-1364\\
         Hungary}
\email{h1175moy\%ella@relay.eu.net}
\thanks{The research of the second author
         was supported by the Hungarian
         National Foundation for Scientific Research,
         under Grant No. 9901.}
```

Example 3   Two authors, same department. I only show the author information section here. The other commands are identical to those in Example 1.
Author information:

```
\author{George~A. Menuhin}
\address{Computer Science Department\\
         University of Winnebago\\
         Winnebago, MN 53714}
\email[George~A. Menuhin]{gmen@ccw.uwinnebago.edu}
\urladdr[George~A. Menuhin]
         {http://math.uwinnebago.edu/homepages/menuhin/}

\thanks{The research of the first author was
         supported by the NSF under grant number~23466.}
\author{Ernest~T. Moynahan}
\email[Ernest~T. Moynahan]{emoy@ccw.uwinnebago.edu}
\thanks{The research of the second author was supported
         by the Hungarian National Foundation for
         Scientific Research, under Grant No. 9901.}
```

Note that the second author has no \address.

Practical Tip 45. The most common mistake in the top matter is the misspelling of a command name; for instance, \adress. LaTeX sends the error message

```
! Undefined control sequence.
l.37 \adress
             {Computer Science Department\\
```

which tells you exactly what you mistyped. Similarly, if you drop a closing brace, as in

```
\email{menuhin@ccw.uwinnebago.edu
```

you are told clearly what went wrong. Because the top matter commands are short (see Section 2.2.3), LaTeX gives the error message

```
Runaway argument?
{menuhin@ccw.uwinnebago.edu \thanks
       {The research of th\ETC.
!File ended while scanning use of \email.
```

If you enclose an optional argument in braces instead of brackets,

```
\title{Complete-simple distributive lattices}
{A construction of complete-simple\\
       distributive lattices}
```

LaTeX uses the short title as the title and the real title is typeset before the title of the typeset article.

Abstract

As we have seen in Section 1.9, you type the abstract in an `abstract` environment, which you place as the last item before the `\maketitle` command. If you place the abstract *after* the `\maketitle` command, LaTeX typesets it wherever it happens to be and sends a warning.

Practical Tip 46. The abstract should be self-contained, so do not include cross-references and do not cite from the bibliography. Avoid user-defined commands.

6.6.5 Options

In the `amsart` document class, there are many options you can use. We list a few.

Font size

Options:	`10pt`	*default*
	`11pt`	
	`12pt`	

This option declares the default font size. You may want to use the `12pt` option for proofreading:

`\documentclass[12pt]{amsart}`

Remember, however, that changing the font size changes the line breaks, so changing the `12pt` option back to `10pt` may require that you make some adjustments in the text (see Section 1.4).

Paper size

Options:	`letterpaper`	(8.5 inches by 11 inches)	*default*
	`legalpaper`	(8.5 inches by 14 inches)	
	`a4paper`	(210 mm by 297 mm)	

Equations and equation numbers

We discuss two options.

Options:	`leqno`	*default*
	`reqno`	

By default, equation numbers are placed on the left, the default `leqno` option. The `reqno` option places the equation numbers on the right.

Option: `fleqn`

This option positions equations a fixed distance from the left margin rather than centering them. The `fleqn` option is typically used in conjunction with the `reqno` option. Here is how an equation looks with the `fleqn` and `reqno` options:

$$\int_{0}^{\pi} \sin x \, dx = 2 \tag{1}$$

typed as

```
\begin{equation}\label{E:firstIntegral}
  \int_{0}^{\pi} \sin x \, dx = 2
\end{equation}
```

Title page

Options: `titlepage`
              `notitlepage`   *default*

The `titlepage` option creates a separate title page including the abstract as in a report. The `notitlepage` option splits the top matter between the first and last pages of the typeset article.

7

Customizing
LATEX

Donald E. Knuth designed TEX as a platform on which *convenient work environments* could be built. One such work environment, LATEX, predominates today, and it is indeed convenient.

Nevertheless, LATEX is designed for all of us, so it is not surprising that we could improve on it for our personal use. There are many reasons to customize LATEX:

Goal 1 to enhance the readability of the source file

Goal 2 to make notational and terminological changes easier

Goal 3 to redefine names used by LATEX

Goal 4 to introduce consistent layouts

Section 7.1 and 7.2 introduce user-defined commands and environments. We dedicate Section 7.3 to the pitfalls of customization. While the benefits of customization are great, there are many practices to avoid.

7.1   *User-defined commands*

LaTeX provides hundreds of commands. Chances are good, however, that you still have specific needs that are not directly addressed by these commands. By judiciously adding *user-defined commands* (or *macros*) you can make your work more productive.

User-defined commands follow the same rules as regular LaTeX commands; see Section 2.2.1.

7.1.1   *Examples and rules*

Commands to enhance readability

Let us start with a few examples of user-defined commands as shorthand for longer command(s) or text in order to enhance readability of the source file (Goal 1).

1. If you use the \leftarrow command a lot, you could define

   \newcommand{\larr}{\leftarrow}

   Then you would only have to type \larr to obtain a left arrow.

2. Instead of

   \widetilde{a}

   you could simply type \wtilda after defining

   \newcommand{\wtilda}{\widetilde{a}}

   I show you how to define a generalized version of such a command in Section 7.1.2.

3. If you want to suppress the ligature in iff (see Section 2.3), you may type

   if\textcompwordmark f

   By defining a command \Iff,

   \newcommand{\Iff}{if\textcompwordmark f}

   you can type \Iff to get iff. We name this command \Iff because \iff is the symbol $\iff$; see Section B.4.

4. If you use the construct $D^{[2]} \times D^{[3]}$ often, you could introduce the \DxD (D times D) command,

   \newcommand{\DxD}{D^{[2]}\times D^{[3]}}

   and then type \DxD instead of the longer, and hard to read, version throughout your document—serves also Goal 2.

5. You can also use commands as a shorthand for text. For instance, if you use the phrase subdirectly irreducible many times in your document, you could define

```
\newcommand{\subdirr}{subdirectly irreducible}
```

`\subdirr` is now shorthand for `subdirectly irreducible`, which typesets as subdirectly irreducible.

Practical Tip 47. If you introduce a command as a shorthand, use the `\xspace` command introduced on page 113.

Practical Tip 48. With modern editors, the need to have user-defined commands as shorthand is reduced. Most editors have "command completion" or "text expansion". For instance, in `TeXShop`, type the first few letters of a word and hit the escape key. The remaining letters are entered to match the first entry in the completion dictionary. To make this feature useful, customize the completion dictionary.

Practical Rule ■ **User-defined commands**

1. Issue the `\newcommand` command.

2. In braces, type the name of your new command, for example, `\subdirr`, including the backslash (`\`).

3. In a second pair of braces, define the command, in this example, `subdirectly irreducible`.

4. Use the command as `\subdirr\`␣ or `\subdirr{}` before a space, before an alphabetical character as `\subdirr{}`, and `\subdirr` otherwise.

Examples for Rule 4. For subdirectly irreducible lattice type
`\subdirr{} lattice`   or   `\subdirr\ lattice`
and not `\subdirr lattice`. Indeed, typesetting `\subdirr lattice` results in subdirectly irreduciblelattice. By the first spacing rule, `\subdirr`␣`lattice` is not any better; (see Section 2.1.1. If you want subdirectly irreducibles, you must use the `\subdirr{}` form. Indeed, `\subdirr{}s` typesets as subdirectly irreducibles.

Using new commands

Practical Tip 49. Place user-defined commands in the preamble of your document.

Practical Tip 50.

- If errors occur, isolate the problem. Comment out the user-defined commands and reintroduce them one at a time.

- LaTeX only checks whether the braces match in the command definition. Other mistakes are found only when the command is used.

For instance, if you define a command with a spelling error

```
\newcommand{\bfA}{\textf{A}}
```

then at the first use of \bfA you get the error message

```
! Undefined control sequence.
\bfA ->\textf
              {A}
```

Note that LaTeX is not complaining about \bfA but about the misspelled \textbf command in the definition of \bfA.

Be careful not to define a user-defined command with a name that is already in use. If you do, you get an error message such as

```
! LaTeX Error: Command \larr already defined.
```

To correct the error, replace the command name with a new one. On the other hand, if you need to replace an existing command, you have to *redefine* it. See Section 7.1.5 for how to do so.

Practical Tip 51. Use spaces to make your source files more readable, but avoid them in definitions.

For example, you can type

```
$D^{ \langle 2 \rangle } + 2 = x^{ \mathbf{a} }$
```

This helps you see how the braces match, easily identify relations and operations, and so on. *Do not add these spaces in command definitions* because it may result in unwanted spaces in your typeset document. You can start a new line to increase the readability of a command definition, provided that you terminate the previous line with %.

Practical Tip 52. In the definition of a new command, command declarations need an extra pair of braces; see Section 2.2.3.

Say you want to define a command that typesets the warning: *Do not redefine this variable!* It is very easy to make the following mistake:

```
\newcommand{\Warn}{\em Do not redefine this variable!}
```

\Warn typesets the warning emphasized, but everything that follows the warning is also emphasized (more precisely, until the end of the \Warn command's scope). Indeed, \Warn is replaced by \em Do not redefine this variable! so the effect of \em goes beyond the sentence to the next closing brace.

The correct definition is

```
\newcommand{\Warn}{{\em Do not redefine this variable!}}
```

The* xspace *package

Rule 4 (on page 111) is the source of many annoying problems in LaTeX. The xspace package helps eliminate such problems. In the preamble, load the package with

`\usepackage{xspace}`

Whenever you define a command that can have such problems, add the \xspace command to the definition. For instance, define \subdirr as

`\newcommand{\subdirr}{subdirectly irreducible\xspace}`

Then all the following typesets `subdirectly irreducible lattice` correctly:

`\subdirr\␣lattice`
`\subdirr{}␣lattice`
`\subdirr␣lattice`

Note that \xspace does not add space if followed by a punctuation mark, so to get

the lattice is subdirectly irreducible.

type

`the lattice is \subdirr.`

Practical Tip 53. Be careful not to use \xspace twice in a definition.

For instance, if you define

`\newcommand{\tex}{\TeX\xspace}`
`\newcommand{\bibtex}{\textsc{Bib}\kern-.1em\tex\xspace}% Bad!!!`

then

`\bibtex, followed by a comma`

typesets as

BIBTEX , followed by a comma

The correct definitions are

`\newcommand{\tex}{\TeX\xspace}`
`\newcommand{\bibtex}{\textsc{Bib}\kern-.1em\TeX\xspace}% Correct!`

Ensuring math

The \ensuremath command is useful for defining commands to work both in text and formulas. Suppose you want to define a command for $D^{\langle 2 \rangle}$. If you define it as

`\newcommand{\Dsq}{D^{\langle2\rangle}}`

then you can use the command in a formula, but not in text. If you define it as

`\newcommand{\Dsq}{$D^{\langle2\rangle}$}`

then it works in text, but not in a formula.

Instead, define this command as

`\newcommand{\Dsq}{\ensuremath{D^{\langle2\rangle}}}`

Then `\Dsq` works correctly in both contexts.

This example also shows the editorial advantages of user-defined commands. Suppose the referee suggests that you change the notation to $D^{[2]}$. To carry out the change you only have to change:

`\newcommand{\Dsq}{\ensuremath{D^{[2]}}}`

It is hard to overemphasize the importance of introducing your notation in user-defined commands. You may want to change notation because:

- you found a better notation

- your coauthor insists

- your article appears in a conference proceedings, and the editor wants to unify the notation

- you are reusing the code from this document in another one, where the notation is different

7.1.2   Arguments

Arguments of user-defined commands work the same way as for LaTeX commands; see page 35. Define

`\newcommand{\fsqAB}{(f^2)^{[[\frac{A^2}{B-1}]]}}`

Then `\fsqAB` typesets as $(f^2)^{[[\frac{A^2}{B-1}]]}$ in a formula. If you use this construct for many functions f, then you may need a generalized command, such as

`\newcommand{\sqAB}[1]{\ensuremath{(#1^2)^{[[\frac{A^2}{B-1}]]}}}`

Now `\sqAB{g}` typesets $(g^2)^{[[\frac{A^2}{B-1}]]}$. The form of this `\newcommand` is the same as before, except that after the name of the command in braces, `{\sqAB}`, we specify the number of arguments in brackets (in this example, `[1]`). Then we can use `#1` in the definition of the command. When the command is invoked, the argument you provide replaces `#1` in the definition.

Notice how these examples disrupt the normal spacing between lines—a practice to avoid!

A user-defined command can have up to nine arguments, numbered 1–9.

Following are some simple examples of user-defined commands with arguments.

1. In the preamble of the source file for this book, I defined

   `\newcommand{\env}[1]{\textnormal{\texttt{#1}}}`

   In this example, the `\env` command is used to typeset environment names. So the environment name center is typed as `\env{center}`. Again the editorial advantage is obvious. If the editor wants the environment names set in sans serif, only one line

in the book has to be changed to alter every occurrence of a typeset environment name:

```
\newcommand{\env}[1]{\textsf{#1}}
```

2. Here is a vector with only one nonzero entry:

$$\langle \dots, 0, \dots, \overset{i}{d}, \dots, 0, \dots \rangle$$

the i above the d indicates that it is the ith component of the vector. A command \vectsup, a vector with a superscript, producing this symbol can be defined as

```
\newcommand{\vectsup}[2]{\langle\dots,0,\dots,
              \overset{#1}{#2},\dots,0,\dots\rangle}
```

\vectsup{i}{d} in a formula now produces $\langle \dots, 0, \dots, \overset{i}{d}, \dots, 0, \dots \rangle$.

7.1.3   Short arguments

There are three ways of defining new commands:

```
\newcommand    \renewcommand    \providecommand
```

We take up the last two in Section 7.1.5. They define commands that can take any number of paragraphs as arguments. The *-ed versions of these commands define *short* commands (see Section 2.2.3) that take a block of text that contains no paragraph break as an argument. For instance,

```
\newcommand*{\bigbold}[1]{{\large\bfseries#1}}
```

makes its short argument large and bold. Short commands are often preferable because of their improved error checking.

7.1.4   Optional arguments

You can define a command whose first argument is *optional,* and provide a *default value* for this optional argument. To illustrate, let us define the command

```
\newcommand{\SimpleSum}{a_{1}+a_{2}+\dots+a_{n}}
```

$\SimpleSum$ now produces $a_1 + a_2 + \cdots + a_n$. Now we change this command so that we can sum from 1 to m if necessary, with n as the default:

```
\newcommand{\BetterSum}[1][n]{a_{1}+a_{2}+\dots+a_{#1}}
```

$\BetterSum$ still produces $a_1 + a_2 + \cdots + a_n$, but $\BetterSum[m]$ typesets as $a_1 + a_2 + \cdots + a_m$.

A \newcommand can have up to nine arguments, but *only the first* may be optional. The following command has two arguments, one optional:

```
\newcommand{\BestSum}[2][n]{#2_{1}+#2_{2}+\dots+#2_{#1}}
```

Now

$\BestSum{a}$	typesets as	$a_1 + a_2 + \cdots + a_n$
$\BestSum{b}$	typesets as	$b_1 + b_2 + \cdots + b_n$
$\BestSum[m]{c}$	typesets as	$c_1 + c_2 + \cdots + c_m$

7.1.5   Redefining commands

LaTeX makes sure that you do not inadvertently define a new command with the same name as an existing command (see, for example, page 112). Assuming that you have already defined the \larr command as in Section 7.1.1 (to typeset ←), to *redefine* \larr, use \renewcommand:

```
\renewcommand{\larr}{\Longleftarrow}
```

and now \larr typesets as ⟸.

💡 **Practical Tip 54.** Use the \renewcommand command sparingly and make sure that you understand the consequences of redefining an existing command. Redefining LaTeX commands can cause LaTeX to behave in unexpected ways, or even crash.

Blind redefinition is the route to madness. See also the discussion in Section 7.3.

You can also use \renewcommand to redefine commands defined by LaTeX or any package. For instance, the end of proof symbol, \qedsymbol, used by the proof environment, can be changed to the solid black square some people prefer (defined in the amssymb package) with the command

```
\renewcommand{\qedsymbol}{$\blacksquare$}
```

Even better, define

```
\renewcommand{\qedsymbol}{\ensuremath{\blacksquare}}
```

so that you can use \qedsymbol in both text and formula. Section 7.1.6 has more on redefining names.

\renewcommand has a companion, \providecommand. If the command it defines has already been defined, the original command is left unchanged. Otherwise, the \providecommand command acts exactly like \newcommand.

7.1.6   Redefining names

A number of names, such as Table, List of Tables, Abstract, and so on, are typeset in your document by LaTeX. You can easily change these names.

For instance, if you are preparing your manuscript for the proceedings of a meeting, and Abstract has to be changed to Summary, you can do so with

```
\renewcommand{\abstractname}{Summary}
```

Table 7.1 lists the commands that define such names, along with their default definitions.

If your document has photographs rather than figures, you could redefine

```
\renewcommand{\figurename}{Photograph}
\renewcommand{\listfigurename}{List of Photographs}
```

\abstractname	Abstract
\appendixname	Appendix
\bibname	Bibliography
\chaptername	Chapter
\contentsname	Contents
\datename	Date
\figurename	Figure
\indexname	Index
\keywordsname	Key words and phrases
\listfigurename	List of Figures
\listtablename	List of Tables
\partname	Part
\proofname	Proof
\refname	References
\see	see also
\seealso	see also
\seeonly	see
\tablename	Table

Table 7.1: Names you can redefine

7.1.7   Localization

What if you want today's date as "25th of November, 2013" and not as "November 25, 2013" as the \today command provides? Use the datetime package by including

\usepackage{datetime}

in the preamble and follow the easy instructions of datetime.pdf to set the date format to what you desire; even to esoteric formats such as "Monday the Twentyfifth of November, Two Thousand Thirteen". You can also put together a date format of your choice using Section 2.3.2.

   Localization transforms the date to the format used in a language other than U.S. English, and not only the date but the millions of features of LaTeX that are language dependent. Localize with the babel package. It magically changes the hyphenation, the quotation marks, the accents, the placement of the accents, and so on.

7.1.8   Defining operators

The powerful \DeclareMathOperator command defines a new operator:

\DeclareMathOperator{\opCommand}{opName}

Invoke the new operator with \opCommand, which is then typeset with opName.

   The \DeclareMathOperator command must be placed in the preamble. For example, to define the operator length (as in Section 1.12), invoked by the command \length, place this in the preamble:

\DeclareMathOperator{\length}{length}

An operator is typeset in math roman with a little space after it, so `$\length A$` typesets as length A. The second argument is typeset as a formula but - and * are typeset as they would be in text. Here are some more examples. Define in the preamble two operators:

```
\DeclareMathOperator{\Trone}{Truncat_{1}}
\DeclareMathOperator{\Ststar}{Star-one*}
```

Then in the body of the article `$\Trone A$` is typeset as $\mathrm{Truncat}_1 A$ and `$\Ststar A$` is typeset as $\mathrm{Star\text{-}one}^* A$.

To define an operator with limits,

```
\DeclareMathOperator*{\doublesum}{\sum\sum}
```

and then (see Section 4.5.4 for multiline subscripts)

```
\[
    \doublesum_{\begin{subarray}{l}
            i^2+j^2 = 50\\
            i,\ j \leq 10
            \end{subarray}}
            \frac{x^i + y^j}{(i + j)!}
\]
```

typesets as

$$\sum_{\substack{i^2+j^2=50\\ i,\ j\leq 10}}\sum \frac{x^i + y^j}{(i + j)!}$$

7.2   User-defined environments

Most user-defined commands are new commands. *User-defined environments,* as a rule, are built on existing environments. We start with such user-defined environments (Section 7.2.1) and then proceed to investigate

- arguments (Section 7.2.2)

- optional arguments (Section 7.2.3)

- short arguments (Section 7.2.4)

Finally, we discuss how to define brand-new environments (Section 7.2.5).

7.2.1   Modifying existing environments

If you do not like the name of the proof environment and would prefer to use the name demo, define

```
\newenvironment{demo}
   {\begin{proof}}
   {\end{proof}}
```

Note that this does not change how the environment is typeset, only the way it is invoked.

To modify an existing environment, `oldenv`, type

```
\com{newenvironment}{name}
   {begin_text}
   {end_text}
```

where `begin_text` contains the command `\begin{oldenv}` and `end_text` contains the command `\end{oldenv}`.

Practical Tip 55. Do not give a new environment the name of an existing command or environment.

For instance, if you define

```
\newenvironment{parbox}
   {...}
   {...}
```

you get the error message

```
! LaTeX Error: Command \parbox already defined.
```

If there is an error in such a user-defined environment, the message generated refers to the environment that was modified, not to your environment. For instance, if you misspell `proof` as `prof` when you define

```
\newenvironment{demo}
   {\begin{prof}}
   {\end{proof}}
```

then *at the first use* of the demo environment you get the message

```
! LaTeX Error: Environment prof undefined.

l.13 \begin{demo}
```

If you define

```
\newenvironment{demo}
   {\begin{proof}\em}
   {\end{prof}}
```

at the first use of demo you get the message

```
! LaTeX Error: \begin{proof} on input line 5
      ended by \end{prof}.

  l.14 \end{demo}
```

Here are two more examples of modified environments.

1. The command

```
\newenvironment{demo}
   {\begin{proof}\em}
   {\end{proof}}
```

   defines a demo environment that typesets an emphasized proof. Note that the scope of \em is the demo environment.

2. The following example defines a very useful environment. It takes an argument to be typeset as the name of a theorem:

```
\newtheorem*{namedtheorem}{\theoremname}
\newcommand{\theoremname}{testing}
\newenvironment{named}[1]{
   \renewcommand{\theoremname}{#1}
      \begin{namedtheorem}}
      {\end{namedtheorem}}
```

   For example,

```
\begin{named}{Name of the theorem}
Body of theorem.
\end{named}
```

   produces

   Name of the theorem. *Body of theorem.*

   in the style appropriate for the \newtheorem* declaration. This type of environment is often used to produce an unnumbered **Main Theorem** or when typesetting a document in which the theorem numbering is already fixed, for instance, when republishing a document in LaTeX.

Redefine an existing environment with the \renewenvironment command. It is similar to the \renewcommand command; see Section 7.1.5.

There are some environments you cannot redefine; for instance, the multiline math environments.

7.2.2 Arguments

An environment defined by the \newenvironment command can take arguments (see Example 2 in Section 7.2.1), but they can only be used in the begin_text argument of the \newenvironment command. Here is a simple example. Define a theorem proclamation in the preamble (see Section 3.3), and then define a theorem that can be referenced:

```
\newenvironment{theoremRef}[1]
   {\begin{theorem}\label{T:#1}}
   {\end{theorem}}
```

This is invoked with

```
\begin{theoremRef}{label}
```

The theoremRef environment is a modified environment. It is a theorem that can be referenced (with the \ref and \pageref commands, of course) and it invokes the theorem environment when it defines T:label to be the label for cross-referencing.

7.2.3 Optional arguments with default values

The first argument of an environment created with the \newenvironment command can be an *optional argument with a default value.* For example,

```
\newenvironment{narrow}[1][3in]
   {\noindent\begin{minipage}{#1}}
   {\end{minipage}}
```

creates a narrow environment. By default, it sets the body of the environment in a 3-inch wide box, with no indentation. So

```
\begin{narrow}
This text was typeset in a \texttt{narrow}
environment, in a 3-inch wide box, with no indentation.
\end{narrow}
```

typesets as

⌐ This text was typeset in a narrow environment, in
  a 3-inch wide box, with no indentation.
└

You can also give an optional argument to specify the width. For example,

```
\begin{narrow}[3.5in]
  This text was typeset in a \texttt{narrow} environment,
  in a 3-inch wide box, with no indentation.
\end{narrow}
```

which produces the following false statement:

⌐ This text was typeset in a narrow environment, in a 3-inch
  wide box, with no indentation.
└

7.2.4   Short contents

We have discussed two commands that define new environments, `\newenvironment` and `\renewenvironment` These commands allow you to define environments whose contents (`begin_text` and `end_text` (see page 119) can include any number of paragraphs. The *-ed versions of these commands define *short* environments whose contents cannot contain a paragraph break (a blank line or a `\par` command).

7.2.5   Brand-new environments

Some user-defined environments are not modifications of existing environments. Here are two examples:

1. A command remains effective only within its scope; see Section 2.2.2. Now suppose that you want to make a change, say redefining a counter, for only a few paragraphs. You could simply place braces around these paragraphs, but they are hard to see. So define

```
\newenvironment{exception}
   {\relax}
   {\relax}
```

and then

```
\begin{exception}
new commands
    body
\end{exception}
```

The environment stands out better than a pair of braces, reminding you later about the special circumstances. The `\relax` command does nothing, but it is customary to include a `\relax` command in such a definition to make it more readable.

2. In this example, we define a new environment that centers its body vertically on a new page:

```
\newenvironment{vcenterpage}
                {\newpage\vspace*{\fill}}
                {\vspace*{\fill}\par\pagebreak}
```

7.3   The dangers of customization

We can customize LaTeX in so many ways. We can add packages to expand its power and define new commands that better suit our work habits. These enhance LaTeX and make it easier to work with. But they also introduce difficulties. Let us start with the obvious.

Whoever introduced the command `\textcompwordmark` knew that—even if we use command completion—we are not going to type `if\textcompwordmark f` to

avoid having a ligature; see Section 2.3. It is a lot of typing, and the source file becomes hard to read. This cries out for a user-defined command, say, \Iff, which is short and *readable* (see Section 7.1.1).

When introducing user-defined commands, watch out for the following traps.

Trap 1. Redefining a command that is a necessary part of LaTeX.

Trap 2. Defining too many commands.

This creates two problems. Your editor has a hard time making changes in your source file. And a few years later, when you want to reuse the material, you have a difficult time understanding all those clever commands.

Trap 3. Your contribution appears in a volume with many other authors and your user-defined commands create conflicts.

As your document appears in a publication, some parts of it are used for the whole volume. The title and maybe even the section titles are used in the Table of Contents.

Practical Tip 56. Do not use your own commands in the title of the document, in the abstract, in section titles, in the bibliography, or in captions of figures and tables.

Practical Tip 57. Do not introduce one-letter commands—for instance, using \C for the complex field—because many one-letter commands are reserved by LaTeX.

Two-letter user-defined commands are not quite this bad. Of the 2,500 or so possibilities only a few dozen are used by LaTeX. The danger here is, of course, conflict with other authors and confusion for the editor. My command file has about 15 two-letter commands. For instance \jj, part of the \jj, \JJ, \JJm family. Also \Id, because Id is the standard notation for ideal lattices. Some editors may think that this is 15 too many.

Practical Tip 58. Do not use \def to define your commands, with the exception of a very few delimited commands.

Using \def means giving up LaTeX's built in defense. In the editorial office of my journal, about half the submitted documents that we cannot typeset violate this rule.

Practical Tip 59. Do not redefine length commands, especially, if you do not know what other length commands are computed based on the ones you change.

Practical Tip 60. Make sure that the packages you use are compatible.

For instance, the popular epsfig and geometry packages cause problems if used with the amsart document class because of the AMS packages.

C H A P T E R

8

Presentations

In Section 1.14, we describe how a *presentation* is a PDF file that you open with Adobe Reader (or some other PDF reader). Then project the presentation one page at a time by pressing the space bar or the arrow keys.

Remember overhead transparencies? If we want to see half of what is on the transparency, we cover up the bottom part so that only the top part is projected. This way we have control over what the audience sees and when. We sometimes used overlays: placing another transparency on top of the projected one to modify it by adding text or graphics.

In this chapter, we further discuss the BEAMER package. We set ourselves a modest goal: using BEAMER with just a few more commands. It is amazing how much you can achieve with a small investment of your time.

8.1 *Baby* BEAMER

In Section 1.14, we learned the \pause command. A frame with a \pause command creates two slides. In BEAMER terminology these pages are *overlays*.

8.1.1   *Overlays*

BEAMER has many commands creating overlays.   Now we discuss \only and
\onslide.

   We introduce overlays with some presentations. The first

```
% babybeamera presentation
\documentclass{beamer}
\begin{document}


\begin{frame}
\frametitle{Some background}


\only<1,2>{We start our discussion with some concepts.}


\only<2>{The first concept we introduce originates
         with Erd\H os.}
\end{frame}
\end{document}
```

   Overlay 1 ignores the second \only command and displays the line as appropriate
to display one line. Overlay 2 displays the two lines as appropriate to display two lines.
As a result, the first line moves slightly up when passing from overlay 1 to overlay 2.
(This is very visible—and unpleasant—when watching a presentation, but harder to
see when printed. To emphasize more the jump, repeat the second line several times.)
The argument of the \only command is typeset only on the overlays specified. On the
other overlays, it is ignored.

   If instead of the \only command you use the \onslide command (on slide, get
it?), as in

```
\onslide<1,2>{We start our discussion with some concepts.}


\onslide<2>{The first concept we introduce originates
         with Erd\H os.}
```

then the first line of overlay 2 completely overlaps the first line of overlay 1, so the first
line seems to stay put.

   The argument of the \onslide command is typeset on the overlays specified and
on the other overlays it is typeset but invisible. This is the behavior you would want
most often, but you may find that sometimes you want \only.

8.1.2   *Understanding overlays*

LaTeX typesets the contents of a frame and the typeset material appears

▪ on all overlays for the parts of the source (maybe all) not modified by any command
  with an overlay specification;

- only on the overlays specified in the arguments of the \only commands;

- on the overlays specified and is typeset but made invisible on the other overlays for the arguments of the \onslide commands.

More on overlay specifications at the end of this section.
Here are some illustrations.

Example 1

```
This is a very \only<1>{very, very} important concept.
\only<1,2>{To start the definition \dots}
```

will typeset overlay 1 as

This is a very very, very important concept. To start the definition . . .

and will typeset overlay 2 as

This is a very important concept. To start the definition . . .

Example 2

```
What is $2+2$? It is \onslide<2>{$4$}.
\only<1>{Can you figure it out?}

\onslide<2>{I hope you all got it right.}
```

will typeset overlay 1 as

What is $2 + 2$? It is . Can you figure it out?

and will typeset overlay 2 as

What is $2 + 2$? It is 4.
I hope you all got it right.

Note that there is space left in overlay 1 for the number 4.

Overlay specifications

The angle brackets contain an *overlay specification*. Here are some more examples:

`<1-2,4->`	means all overlays from 1 to 2, and all overlays from 4 onwards
`<-3>`	means all overlays up to 3
`<2,4,6>`	means overlays 2, 4, and 6

In the presentation `babybeamera` (in the `samples` folder) we have two overlay specifications: `<1,2>` and `<2>`. Maybe, `<1->` and `<2->` would be better, so that if you add a third overlay you do not have to change these.

8.1.3   Lists as overlays

Lists can be presented one item at a time, for example the `babybeamerb` presentation in Figure 8.1 (in the `samples` folder) shows the three overlays of a list. R. Padmanabhan appears on the first, R. Padmanabhan and Brian Davey appear on the second, and so on. This is accomplished simply by adding the overlay specification `<1->` to the item for R. Padmanabhan, the overlay specification `<2->` to the item for Brian Davey, and so on.

```
% babybeamerb presentation
\documentclass{beamer}
\begin{document}

\begin{frame}
\frametitle{Overlaying lists}
```


Figure 8.1: `babybeamerb` presentation

```
We introduce our guests:
\begin{itemize}
\item<1-> R. Padmanabhan
\item<2-> Brian Davey
\end{itemize}
\end{frame}
\end{document}
```

Such an overlay structure is used so often that BEAMER has a shorthand for it, [<+->]. Here it is in babybeamerc (in the samples folder).

```
% babybeamerc presentation
\documentclass{beamer}
\begin{document}

\begin{frame}
\frametitle{Overlaying lists}

We introduce our guests:
\begin{itemize}[<+->]
\item R. Padmanabhan
\item Brian Davey
\item Harry Lakser
\end{itemize}
\end{frame}
\end{document}
```

This shorthand allows adding and reordering items without having to change overlay specifications.

Of course, if you do not want the items to appear in sequence, you have to use overlay specifications.

8.1.4 *Out of sequence overlays*

We now present an example of "out of sequence overlays". Look at Figure 8.2. I want to make this part of my presentation. First, I want to show the theorem, then illustrate it with the diagram at the bottom. Finally, I present the proof in the middle. So I need three overlays.

The theorem is on all three overlays, 1, 2, 3. Its illustration is on overlays 2 and 3, leaving room for the proof that appears only on overlay 3.

This is an example of "out of sequence overlays". We code this in babybeamerd (in the samples folder).

Since declarations, proofs, and the \includegraphics command can all have overlay specifications, this is easy to accomplish.

```
% babybeamerd presentation, first try
\documentclass{beamer}
\begin{document}


\begin{frame}
\frametitle{Overlaying declarations and graphics}
\begin{theorem}<1->
Every finite distributive lattice can be embedded
in a boolean lattice.
\end{theorem}
\begin{proof}<3->
Use join-irreducible elements.
\end{proof}
\onslide<2->{\includegraphics{cube}}
\end{frame}
\end{document}
```

8.1.5   Blocks and overlays

You can think of a theorem in BEAMER as the contents of the theorem environment with a heading and, optionally, with an overlay specification, and with most themes—see Section 8.4—colorful visual highlighting.

BEAMER provides the block environment that works the same way except that you name the block. The (partial) syntax of the block environment is

```
\begin{block}<overlay spec>{title}
    source
\end{block}
```

Blocks are shaped as theorems. If there is no title, you still need the braces. The overlay specification is optional.
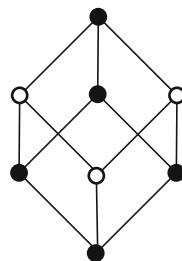
Figure 8.2: The slide to represent

8.1.6   Links

Some links are automatically provided. The sidebar of a slide is a *navigation bar.* First, it shows which section you are in. Second, clicking on a section title takes you to that section.

Creating your own link is a two-step process.

1. Name the place you want to link to.

2. Create a button with the property that clicking on it jumps you to the designated place.

To illustrate this process, open the file `babybeamerc.tex` and save it in the `work` folder as `babybeamere.tex`.

1. Name the frame you want to link to by adding a label to the `\begin{frame}` line. In `babybeamere`, add a label to the frame `threeguests`:

    ```
    \begin{frame}[label=threeguests]
    ```

    Labels of frames are also useful for selective typesetting of your presentation; see Section 8.5.

2. Add the following line to `babybeamere`:

    ```
    \hyperlink{threeguests<3>
    {\beamergotobutton{Jump to third guest}}
    ```

    This creates a link to the third overlay of the frame named `threeguests`, and creates a button, with the text `Jump to third guest`. Clicking on this button will jump to the third overlay of the frame `threeguests`.

3. To add variety to linking, include a new first frame:

    ```
    \begin{frame}
    \frametitle{First frame with a button}
    Button example

    Jumping to an overlay of a different frame
    \bigskip

    \hyperlink{threeguests<3>}
    {\beamergotobutton{Jump to third guest}}
    \end{frame}
    ```

    which has a button for jumping to the third overlay of the `fourguests` frame.

4. We also add a new third frame.

    ```
    \begin{frame}
    \frametitle{Third frame with a button}
    ```

```
Button example

Jumping to another frame
\bigskip

\hyperlink{threeguests}
{\beamergotobutton{Jump to guest list}}
\end{frame}
```
with a button, with the text `Jump to guest list`. Clicking on this button will jump to the second frame, overlay not specified (defaults to 1).

5. Add a fourth frame,
```
\begin{frame}
\frametitle{Hidden link}
\hyperlink{threeguests}{Jumping to the guest list}
\end{frame}
```
introducing another version of the \hyperlink command:
`\hyperlink{threeguests}{Jumping to the guest list}`
which typesets the second argument as regular text, making it an *invisible link*. However, you may notice that the cursor changes when it hovers over the link. For instance, you may want to link the use of a concept to its earlier definition, where you also need a button for the return jump.

Here is babybeamere:
```
% babybeamere presentation
\documentclass{beamer}
\begin{document}

\begin{frame}
\frametitle{First frame with a button}

Button example

Jumping to an overlay of a different frame
\bigskip

\hyperlink{threeguests<3>}{\beamergotobutton{Jump to
                                          third guest}}
\end{frame}
\begin{frame}[label=threeguests]
\frametitle{Overlaying lists}
```

Figure 8.3: `babybeamere` presentation

```
We introduce our guests:
\begin{itemize}
\item<1-> R. Padmanabhan
\item<2-> Brian Davey
\item<3-> Harry Lakser
\end{itemize}


\hyperlink{threeguests<3>}{\beamergotobutton{Jump to third guest}}
\end{frame}

\begin{frame}
\frametitle{Third frame with a button}
Button example

Jumping to another frame
\bigskip

\hyperlink{threeguests}{\beamergotobutton{Jump to guest list}}
\end{frame}

\begin{frame}
\frametitle{Hidden link}

\hyperlink{threeguests}{Jumping to the guest list}
\end{frame}
\end{document}
```
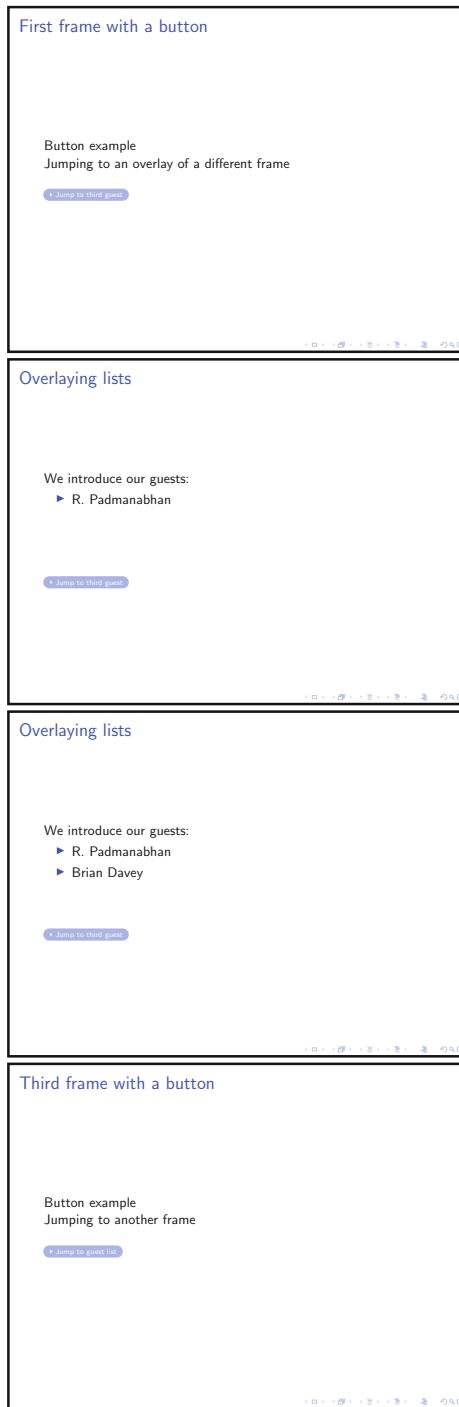
Figure 8.3 shows all these buttons. We do not show overlays 3 and 4 of frame 2 and frame 4, where the button is invisible.

8.1.7   Columns

It is often useful to put the display into columns. A simple illustration is given in babybeamerf:

```
% babybeamerf presentation
\documentclass{beamer}
\begin{document}

\begin{frame}
\frametitle{Columns, top alignment}
```

```
\begin{columns}[t]
\begin{column}{2in}
Is it true that there is no new result
on the Congruence Lattice Characterization Problem?
\end{column}
\begin{column}{2in}
F. Wehrung found a distributive algebraic lattice that
cannot be represented as the congruence lattice
of a lattice.
\end{column}
\end{columns}
\end{frame}
\end{document}
```

The environment is columns. It has an optional argument for alignment, t for top, c for center, and b for bottom.

The columns, usually two, are both in the column environment; the width of the column is in the argument; it can be given as a measurement—2in in the example—or relative to the width of the whole frame as 0.4\textwidth.

Figure 8.4 shows the babybeamerf presentation.
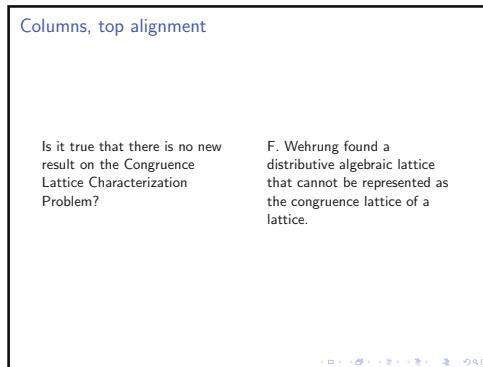
Figure 8.4: babybeamerf presentation

8.1.8 *Coloring*

LaTeX's job is to produce documents that contain text, formulas, and graphics. Such documents are, as a rule, not published in color. Presentations are different. If you prepare a color presentation, it will project in color.

Nevertheless, the color commands are of limited use even for presentations. You will probably use the color scheme of the chosen theme (see Section 8.4), and have limited opportunity to color things yourself.

Practical Tip 61. Too much color distracts from the presentation but judicious use of color—say, for highlighting a word or phrase—can be very effective.

BEAMER has seventeen predefined colors: red, green, blue, cyan, magenta, yellow, orange, violet, purple, brown, pink, olive, black, darkgray, gray, lightgray, and white. With the proper options, there are hundreds more. So you can color text green with the command

```
\textcolor{green}{This text is green.}
```

To pretty things up, you can use

```
\colorbox{green}{Green box}
```

which puts the argument in a green box and

```
\fcolorbox{red}{green}{Green box}
```

which also adds a red frame.

8.2   The structure of a presentation

The structure of your presentation is, by and large, determined by the sectioning commands: \section and \subsection. For a very long lecture there may also be \part commands. The argument of any of these commands can have a short version for the navigational side bar; see Section 8.1.6.

The sectioning commands used in a BEAMER presentation look the same as they do for documents, but they play a different role: adding an entry to the Table of Contents. They also act as place markers in the sense that if you click on the title of a section in a navigation bar, then you will jump to the *frame following* the section command.

Practical Rule ■ Sectioning commands

1. Sectioning commands can only be placed between frames.

2. There must be a frame following the last sectioning command.

3. For a long (sub)section title, use \breakhere to break a line.

4. The optional short versions are for the navigation bar.

These are illustrated with `beamerstructure1` (in the `samples` folder), see Figure 8.5. The line

```
\tableofcontents[pausesections, pausesubsections]
```

causes the Table of Contents to appear a line at a time. This command can also be used without an option or with only one, `pausesections`.

The second slide shown in Figure 8.5 is the Table of Contents.

```
% beamerstructure1 presentation
\documentclass{beamer}
\usetheme{Berkeley}
\begin{document}

\begin{frame}
\frametitle{Outline}

\tableofcontents[pausesections, pausesubsections]
\end{frame}
```


Figure 8.5: `beamerstructure1` presentation, slides 3 and 5

```
\section[Sec1]{Section 1}

\begin{frame}
\frametitle{Section 1}

Text of Section 1
\end{frame}

\subsection[Sec1 Subsec1]{Section 1 -- Subsection 1}

\begin{frame}
\frametitle{Section 1\\Subsection 1}

Text of Section 1, Subsection 1
\end{frame}

\subsection[Sec1 Subsec2]{Section 1 -- Subsection 2}
\begin{frame}
\frametitle{Section 1\\Subsection 2}

Text of Section 1, Subsection 2
\end{frame}

\subsection[Sec1 Subsec3]{Section 1 -- Subsection 3}
\begin{frame}
\frametitle{Section 1\\Subsection 3}

Text of Section 1, Subsection 3
\end{frame}

\section[Sec2]{Section 2}

\begin{frame}
\frametitle{Section 2}

Text of Section 2
\end{frame}

\end{document}
```
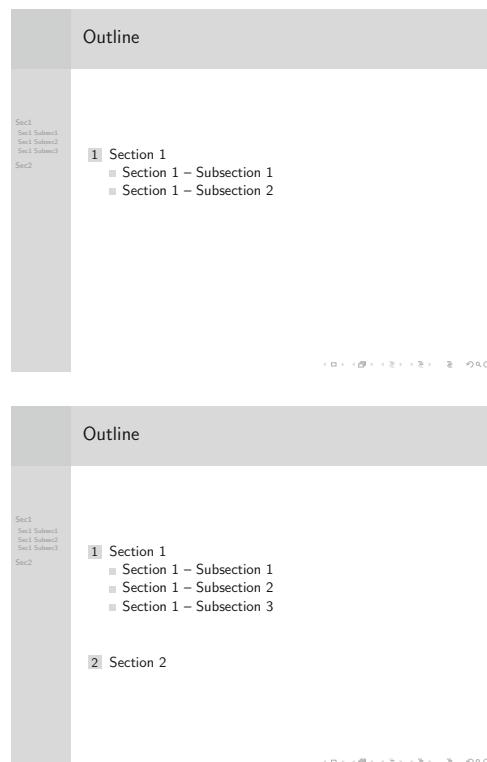
8.2.1   Longer presentations

Longer presentations may need parts and a more complicated Table of Contents. I will not discuss these topics, but the presentation `beamerstructure2` (in the `samples` folder) illustrates the use of parts and some other features. I added some comments to point these out. See Figure 8.6 for two sample pages of this presentation.
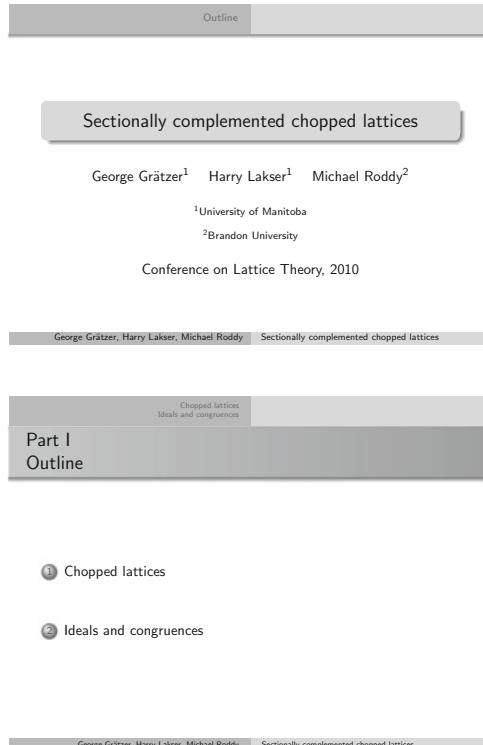
Figure 8.6: `beamerstructure2` presentation, slides 1 and 2

8.2.2   Navigation symbols

The more structure you have in a presentation, the more you may appreciate the navigation icons shown by default on each page in the last line on the right. The icons are:

- the slide

- the frame

- the section

- the presentation icons

each surrounded by a left and a right arrow

- the appendix

- the back and forward icons (circular arrows)

- the search icon (a magnifying glass)

If you decide not to have them, as in the presentation `beamerstructure2`, then give the following command in the preamble:

`\setbeamertemplate{navigation symbols}{}`

8.3   Notes

You can place notes in your presentation to remind yourself of what you want to say in addition to what is being projected. A note is placed in the presentation as the argument of the `\note` command, as in

`\note{This is really difficult to compute.}`

By default, notes are not shown in the presentation. If you invoke BEAMER with

`\documentclass[notes=show]{beamer}`

then the notes pages are included. The command

`\documentclass[notes=show, trans]{beamer}`

produces transparencies with notes, and

`\documentclass[notes=only]{beamer}`

produces only the note pages, one note page for every overlay of a frame with a note. To avoid this, print the output of

`\documentclass[trans, notes=only]{beamer}`

In addition to these examples, all the notes placed in a single frame are collected together on one note page. And a note between frames becomes a page on its own.

BEAMER does an excellent job of producing notes pages, for an example, see Figure 8.7. In the upper-left corner, it displays precisely where we are in the structure of the presentation. The upper-right corner shows a small picture of the page to which the notes are attached.

8.4   Themes

You can achieve detailed control over your presentation by defining all the elements yourself. BEAMER places dozens of commands at your disposal to make this possible. Or you can use a *presentation theme* that will do the job for you.

Figure 8.7: A note page

The command to name a presentation theme is \usetheme{}. The presentation beamerstructure2 uses the theme Warsaw (see Figure 8.6), so following the document class line type the command

```
\usetheme{Warsaw}
```

quickbeamer1 uses the theme Berkeley (see Figure 8.5).

The presentation themes are in the theme subfolder of the themes folder of BEAMER. A presentation theme defines all the colors, but you can alter them with the command \usecolortheme{}. You have a choice of albatross, beetle, crane, fly, and seagull.

For instance,

```
\usetheme{Warsaw}
\usecolortheme{seagull}
```

is a gray version of the Warsaw theme, appropriate for printing in black-and-white.

8.5   *Planning your presentation*

Step 1 As a rule, your presentation is based on one or more of your documents. Collect them in one folder. Resolve naming conventions as necessary. There should be only one Fig1!

Step 2 Rewrite the document(s) to sketch out your presentation. The pages correspond to frames. A page should not have too many words, say, no more than 40. Replace your numbered theorems with named theorems. Never reference another page. Have few sections and subsections. Add a Table of Contents, which is a readable overview of the new document.

Step 3 Base the new presentation on a presentation in the samples folder, a sample presentation in BEAMER's solution folder, or on one of your own or of a colleague's older presentations. Turn the pages into frames.

Step 4 Design your frames and add frame titles. You are completely responsible for the visual appearance of every frame and overlay.

This is, of course, in addition to brevity and readability. Do not let LaTeX break your lines. Do it with the \\ command and keep words that belong together on the same line.

Step 5 Write notes to remind yourself what you want to say in your lecture that is not on the slides. Print the notes for your lecture.

Step 6 Build in flexibility. For instance, if you have four examples to illustrate a definition, put each one on a different frame or overlay, and add a link to each that skips the rest of the examples. Depending on your audience's understanding, show an example or two, and skip the rest. The same way, you may skip proof ideas and even topics.

Step 7 Prepare for the worst—the computer system may fail, but projectors seldom do—so print a set of transparencies for your lecture as a backup by invoking the option `trans` of the documentclass
`\documentclass[trans]{beamer}`

To print a *handout*, use the `handout` option
`\documentclass[handout]{beamer}`

Open the presentation in Acrobat Reader. In `Printer/Page Setup...` set landscape and 140 magnification. In the `Print` dialogue box in `Layout` choose two pages per sheet and print—assuming, of course, that you have a printer offering these options.

8.6   What did I leave out?

Since the BEAMER reference manual is 245 pages, it is clear that this chapter covers maybe 10% of it.

For many presentations, even Section 1.14 will do.

You can do very simple animation with what we have covered here. This is illustrated with the `babybeamerg` presentation (in the `samples` folder).

```
% babybeamerg sample document
\begin{document}

\begin{frame}

\includegraphics<1>{basem3-1}

\includegraphics<2>{basem3-2}

\includegraphics<3>{basem3-3}
```

```
\includegraphics<4>{basem3-4}
\end{frame}
\end{document}
```

The congruence generated by the dashed red line, see Figure 8.8, spreads in three steps, illustrating an interesting result. The animation is quite effective and instructive.

If you want to place such changing pictures lower in a frame, put them in the overprint environment.
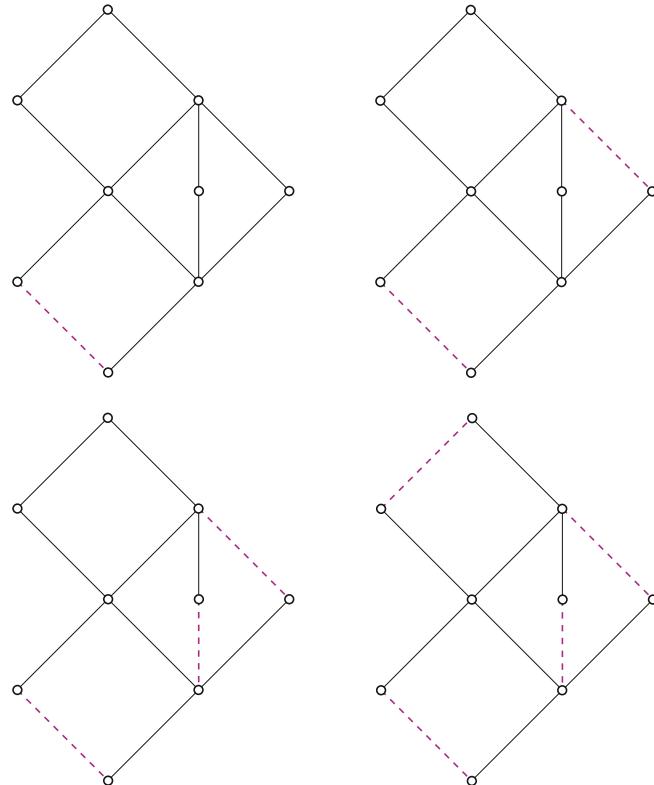
Figure 8.8: The four pictures of babybeamerg

Illustrations

Many documents require illustrations. We discussed in Section 1.11 how to include a PDF file as an illustration. Simpler illustrations—using circles and lines—you can easily do with Till Tantau's Ti*k*Z package. The few commands we discuss in this chapter may serve your needs. If you want to learn more, go to the Ti*k*Z manual, it covers a lot more in 726 pages.

This chapter is based on Jacques Crémer's *A very minimal introduction to Ti*k*Z,* with his permission, and Michael Doob's detailed suggestions.

9.1 Your first picture

To use the Ti*k*Z package, include

```
\usepackage{tikz}
```

in the preamble of your document. A picture is in a `tikzpicture` environment, which is, in turn, typically within a `figure` environment (see Section 1.11):
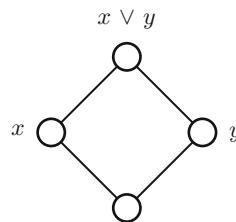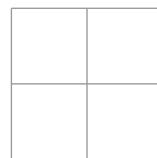
Figure 9.1: Our first Ti*k*Z illustration

```
\begin{figure}[htb]
{\centering
\begin{tikzpicture}
...
\end{tikzpicture}}
\end{figure}
```

We draw the illustration of Figure 9.1.

Step 1: Draw the grid
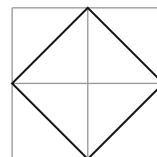
The command is \draw, the optional argument: help lines.

```
\begin{tikzpicture}
\draw[help lines] (0,0) grid (2,2);
\end{tikzpicture}
```

Note the semicolon terminating the line. If you forget, you get the helpful reminder:

```
Package tikz Error: Giving up on this path.
Did you forget a semicolon?
```

Step 2: Draw the four edges

The command is \draw; the argument is a series of grid points connected by --
(two dashes).

```
{\centering\begin{tikzpicture}
\draw[help lines] (0,0) grid (2,2);
\draw (1,0)--(2,1)--(1,2)--(0,1)--(1,0);
\end{tikzpicture}}
```

Again, note the semicolon terminating the line. It has to terminate all Ti*k*Z lines!!! No
more warnings.

Let me specify the conventions used in my field for such a diagram (note that
1 inch is 2.54 cm and 1 cm is 28.35 points): We use a grid with lines 1 cm apart; the
circles have radius 1.8 mm and line width 1 pt; the lines have line width 0.7 pt.

So the \draw command we would use for the illustration is

```
\draw[line width=0.7pt] (1,0)--(2,1)--(1,2)--(0,1)--(1,0);
```

to make the lines a little thicker. More about line width soon.

Step 3: Draw the circles
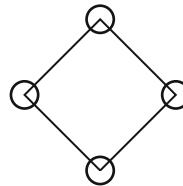
We add a circle with the \draw command

```
\draw (1,0) circle[radius=1.8mm];
```

By the conventions (above), we want the line width to be 1pt:

```
\draw[line width=1pt] (1,0) circle[radius=1.8mm];
```

We add four circles at (1,0), (2,1), (1,2), (0,1):

Step 4: Fill the circles

This looks ugly. We should not be seeing the line segments in the circles. Easy to help.
Fill the circles with white by adding the fill=white option to \draw. So we get the
illustration of Figure 9.1 with the grid. We comment out the line producing the grid:

```
{\centering\begin{tikzpicture}
%\draw[help lines] (0,0) grid (2,2);
\draw[line width=0.7pt](1,0)--(2,1)--(1,2)--(0,1)--(1,0);
\draw[fill=white, line width=1pt] (1,0) circle[radius=1.8mm];
```

```
\draw[fill=white, line width=1pt] (2,1) circle[radius=1.8mm];
\draw[fill=white, line width=1pt] (1,2) circle[radius=1.8mm];
\draw[fill=white, line width=1pt] (0,1) circle[radius=1.8mm];
\end{tikzpicture}}
```

and this produces the illustration of Figure 9.1 except for the labels.

Step 5: Add the labels

We do this with the \node at command. To add the label y to the circle with center at (2,1):

```
\node at (2.5,1) {$y$};
```

You get 2.5 by $2.5 = 2 + 0.18 +$ a little nudge. Experiment until you like the result. Then proceed to the other circles:

```
\begin{figure}[h!]
{\centering\begin{tikzpicture}
%\draw[help lines] (0,0) grid (2,2);
\draw[line width=0.7pt] (1,0)--(2,1)--(1,2)--(0,1)--(1,0);
\draw[fill=white, line width=1pt] (1,0) circle[radius=1.8mm];
\draw[fill=white, line width=1pt] (2,1) circle[radius=1.8mm];
\node at (2.5,1) {$y$};
\draw[fill=white, line width=1pt] (1,2) circle[radius=1.8mm];
\node at (1,2.5) {$x \vee y$};
\draw[fill=white, line width=1pt] (0,1) circle[radius=1.8mm];
\node at (-0.5,1) {$x$};
\end{tikzpicture}}
\caption{Our first \tikzname illustration}\label{Fi:firsttikz}
\end{figure}
```

producing the illustration of Figure 9.1!

Step 6: Remember, this is LATEX

Section 7.1 introduced user-defined commands. What better place to use it? Define

```
\newcommand{\mycircle}[1]
{\draw[fill=white,line width=1pt] (#1) circle[radius=1.8mm]}
```

Even better. TikZ allows us to set default values. The command

```
\tikzset{every picture/.style={line width=0.7pt}}
```
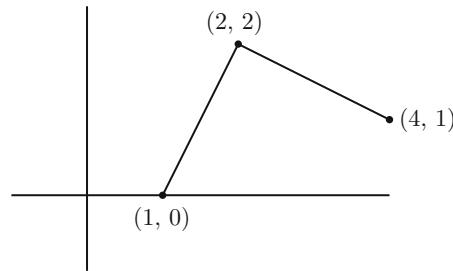
sets the default value of line width to 0.7pt. Now the code for Figure 9.1 becomes easier to read (and write):

```
{\centering\begin{tikzpicture}
%\draw[help lines] (0,0) grid (2,2);
\draw (1,0)--(2,1)--(1,2)--(0,1)--(1,0);
\mycircle{1,0};
\mycircle{2,1};
\node at (2.5,1) {$y$};
\mycircle{1,2};
\node at (1,2.5) {$x \vee y$};
\mycircle{0,1};
\node at (-0.5,1) {$x$};
\end{tikzpicture}}
```
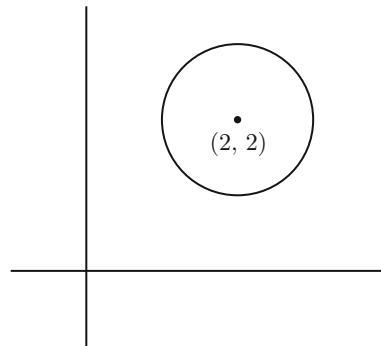
9.2   The building blocks of an illustration

An illustration is built from components. We discuss some of them: line segments, circles, dots (or vertices), ellipses, rectangles, arcs, smooth curves (Bézier curves), and labels.

Line segments A path drawn with the command \draw (1,0)--(2,2)--(4,1);
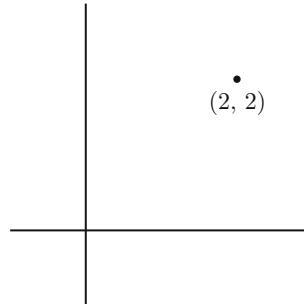
Circles A circle drawn using \draw (2,2) circle[radius=1];
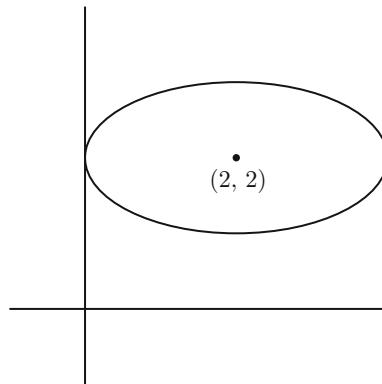
Dots (or vertices) The command
```
\draw[fill] (2,2) circle[radius=1pt];
```
 draws a dot (or vertex):
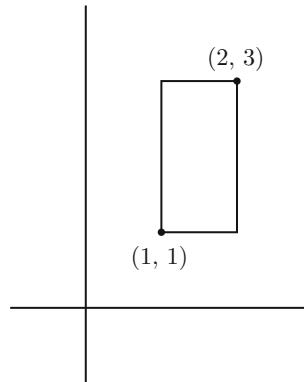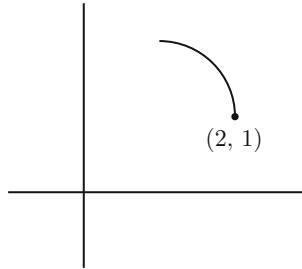
Ellipses Drawn by `\draw (2,2) ellipse[x radius=2, y radius=1];`

Rectangles A rectangle drawn with `\draw (1,1) rectangle (2,3);`

Arcs `\draw (2,1) arc[start angle=0, end angle=90, radius=1];`
draws an arc of a circle:
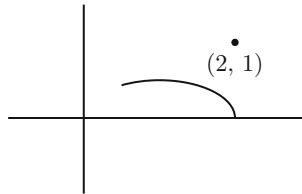
Surprise! The point is not the center of the circle.

Finally, an arc of an ellipse is drawn with the command

```
\draw (2,0) arc[x radius=1cm, y radius=5mm,
           start angle=0, end angle=120];
```

which typesets as
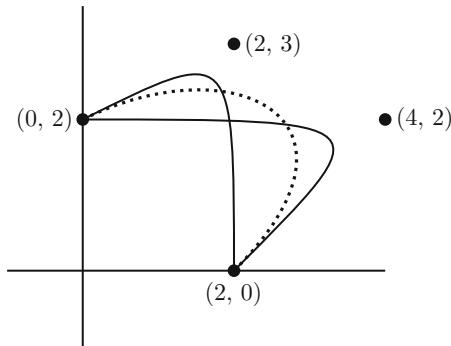
Smooth curves (Bézier curves) Nice curves can be drawn with a single control point
(quadratic), as in

```
\draw (2,0)..controls (2,3)..(0,2);
\draw (2,0)..controls (4,2)..(0,2);
```
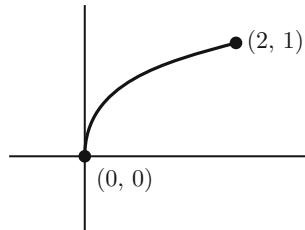
or with two control points (cubic):

```
\draw[dotted] (2,0)..controls (4,2) and (2,3)..(0,2);
```
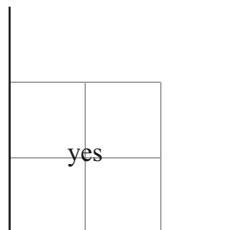
These three curves typeset as

Alternatively, draw the curve defined by two points, A and B, the start and the end, and by the direction it leaves A and the direction it arrives at B with the command: `\draw[very thick] (0,0) to[out=90,in=195] (2,1.5);` This draws a curve from `(0,0)` to `(2,1)` which "leaves" at an angle of $90°$ and "arrives" at an angle of $195°$:
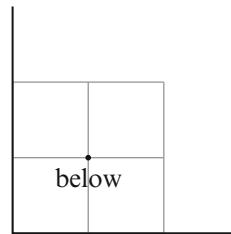
Labels   We place text or formula in a picture with `\node at (1,1) {yes};`

Notice how the "yes" is positioned relative to (1,1).

To place a label *below* a point, use the option `below`:
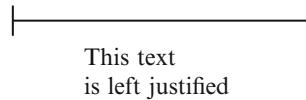
with the command `\node[below] at (1,1) {below};`. You can also use `above`, `left`, and `right`, and their combinations, for instance, `above left`.

If the text is several lines long, break it with `\\` and tell Ti*k*Z how to align it:

```
\begin{mypicture}[xscale=1.3]
\draw[thick] (0,0)--(3,0);
\draw (0,-.2)--(0,.2);
\draw (3,-.2)--(3,.2);
\node[align=left, below] at (1.5,-.5)%
    {This text\\ is left justified};
\end{mypicture}
```

which typesets as

which typesets as

Text could be

- left justified, option: `align=left`;

- right justified, option: `align=right`;

- centered, option: `align=center`;

9.3   Transformations

We can rotate, translate, and scale the illustrations.
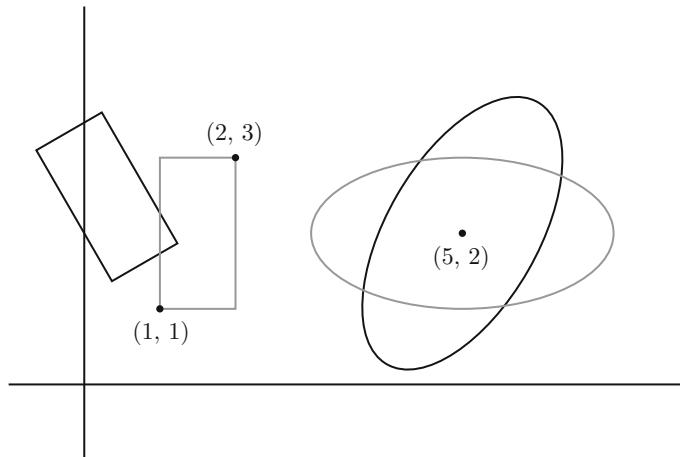
Rotations The command
    `\draw[rotate=30] (1,1) rectangle (2,3);`
    rotates the rectangle by $30°$ (around the origin) and
    `\draw (2,2) ellipse[x radius=2, y radius=1, rotate=60];`
    rotates an ellipse by $60°$ (around its center):
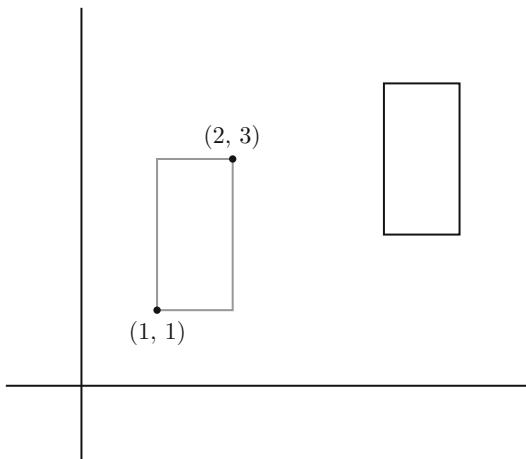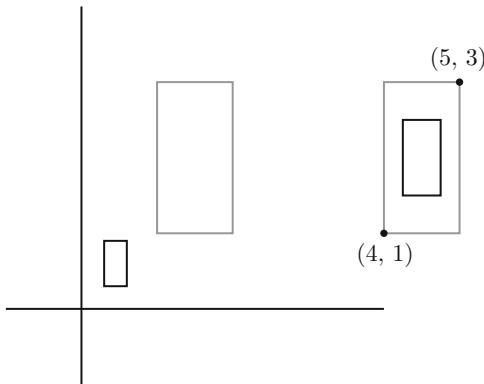
Translations `\draw[shift={(3,1)}] (1,1) rectangle (2,3);` shifts the rectangle:

Scaling The command `\draw[scale=0.3] (1,1) rectangle (2,3);` scales the rectangle by 0.3 from the origin and

```
\draw[scale around={0.5:(4.5,2)}] (4,1) rectangle (5,3);
```
scales the rectangle by 2.5 using around the center of the rectangle, that is, around $(4.5, 2)$:

You can scale the two dimensions in different proportions:

coded with

```
\begin{tikzpicture}[xscale=1.5, yscale=0.5]\\
\draw (2,2) circle[radius=1.5];\\
```

or with

```
\draw[xscale=1.5, yscale=0.5] (2,2) circle[radius=1.5];
```

9.4   Path attributes

A `\draw` command draws a path, with a start point and an end point. The start and end points of `\draw (1,0)--(2,1)--(1,2)--(0,1);` are `(1,0)` and `(0,1)`; of `\draw (2,2) circle[radius=1.5];` are `(2,2)` and `(2,2)`.

We consider now some of the common attributes.

Line width We have already seen the `line width=1pt` option of `\draw`. Ti*k*Z comes with seven additional built in widths: `ultra thin`, `very thin`, `thin`, `semithick`, `thick`, `very thick`, and `ultra thick`.

Dashes and dots You can also make dotted and dashed lines. The commands

```
\draw[dashed] (0,0.5)--(2,0.5);
\draw[dotted, thick] (0,0)--(2,0);
```

make two thick lines, one dashed and one dotted:

- - - - - - - - - -

....

Colors Articles, as a rule, are printed black and white. But illustrations in PDF files and in presentations are shown in full color.

Ti*k*Z comes with the following colors ready to use: red, green, blue, cyan, magenta, yellow, black, gray, darkgray, lightgray, brown, lime, olive, orange, pink, purple, and teal.

The following example uses white, lightgray, and gray.

Arrows We can put arrows or bars on one or both ends of a path:

coded as

```
\begin{tikzpicture}
\draw[->, lightgray] (0,0)--(3,0)--(3,-1);
\draw[<-] (0,-0.5)--(2,-0.5);
\draw[|->, dashed] (0,-1)--(2,-1);
\draw[very thick, <->] (0,-3) to[out=90,in=195] (3,-1.5);
\end{tikzpicture}
```
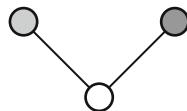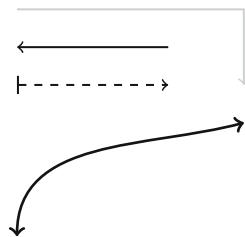
Ti*k*Z provides you with dozens of arrows. You get them by invoking the arrows library with \usepgflibrary{arrows} in the preamble. Even better, use the tikz-cd package by F. Neves invoked by \usepackage{tikz-cd}. It is designed to code commutative diagrams and it provides arrows very close to the LaTeX style, see Figure 9.2. It comes with an excellent (and short) manual tikz-cd-doc.pdf.

`rightarrow`	yields $\longrightarrow$
`leftarrow`	yields $\longleftarrow$
`leftrightarrow`	yields $\longleftrightarrow$
`dash`	yields $\longrightarrow\!\!\!\!-$
`Rightarrow`	yields $\Longrightarrow$
`Leftarrow`	yields $\Longleftarrow$
`Leftrightarrow`	yields $\Longleftrightarrow$
`equal`	yields $=\!=\!=$
`mapsto` (or `maps to`)	yields $\longmapsto$
`mapsfrom`	yields $\longleftarrow\!\mid$
`hookrightarrow` (or `hook`)	yields $\hookrightarrow$
`hookleftarrow`	yields $\hookleftarrow$
`rightharpoonup`	yields $\longrightharpoonup$
`rightharpoondown`	yields $\longrightharpoondown$
`leftharpoonup`	yields $\longleftharpoonup$
`leftharpoondown`	yields $\longleftharpoondown$
`dashrightarrow` (or `dashed`)	yields $\dashrightarrow$
`dashleftarrow`	yields $\dashleftarrow$
`rightarrowtail` (or `tail`)	yields $\rightarrowtail$
`leftarrowtail`	yields $\leftarrowtail$
`twoheadrightarrow` (or `two heads`)	yields $\twoheadrightarrow$
`twoheadleftarrow`	yields $\twoheadleftarrow$
`rightsquigarrow` (or `squiggly`)	yields $\rightsquigarrow$
`leftsquigarrow`	yields $\leftsquigarrow$
`leftrightsquigarrow`	yields $\leftrightsquigarrow$

Figure 9.2: The arrows provided by the `tikzcd` package

In the `tikzcd` environment, the command `\arrow` produces an arrow. It takes one argument, a character `r`, `l`, `u`, or `d`, for right, left, up and down. A label is placed on an arrow as the second argument.

Here are two examples of commutative diagrams from the `tikz-cd` manual.
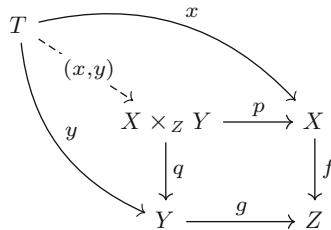
A basic example:

$$
\begin{array}{ccc}
A & \xrightarrow{\psi} & B \\
\downarrow & & \downarrow{\scriptstyle\psi} \\
C & \xrightarrow{\eta} & D
\end{array}
$$

coded as

```
\begin{tikzcd}
  A \arrow{r}{\psi} \arrow{d}
    & B \arrow{d}{\psi} \\
  C \arrow{r}{\eta}
    & D
\end{tikzcd}
```

An example with curved and dotted arrows:

$$
\begin{array}{ccc}
T & \xrightarrow{x} & \\
& (x,y) & \\
& X \times_Z Y \xrightarrow{\;p\;} X & \\
y & \big\downarrow q \qquad\quad \big\downarrow f & \\
& Y \xrightarrow{\;g\;} Z &
\end{array}
$$

```
\begin{tikzcd}
  T
  \arrow[bend left]{drr}{x}
  \arrow[bend right]{ddr}{y}
  \arrow[dashed]{dr}[description]{(x,y)} & & \\
    & X \times_Z Y \arrow{r}{p} \arrow{d}{q}
      & X \arrow{d}{f} \\
    & Y \arrow{r}{g}
      & Z
\end{tikzcd}
```

9.5   *What did I leave out?*

Since the Ti*k*Z manual is 726 pages, it is clear that this chapter covers maybe 2% of it. For most math illustrations, this chapter will do. (I use Adobe Illustrator for my lattice diagrams. I use maybe 2% of Illustrator's power for my work.)

Ti*k*Z can plot. It can graph many built in functions, has a small programming language, has libraries, for instance, for circuits. Figure 9.3 shows an example of a circuit by Erno Pentzin.

Figure 9.4 is by Daniel Steger, a decorative element from a mosaic in the living room of Casa degli Armorini Dorati, Pompeii. The example shows the power of the Ti*k*Z mathematical engine. Our final example, on page 161, is from the Ti*k*Z manual, a course outline. Imagine the last two in full color!

You find the code, code.tex, for Figures 9.3 and 9.4 in the samples folder. For Figure 9.5, see the Ti*k*Z manual.
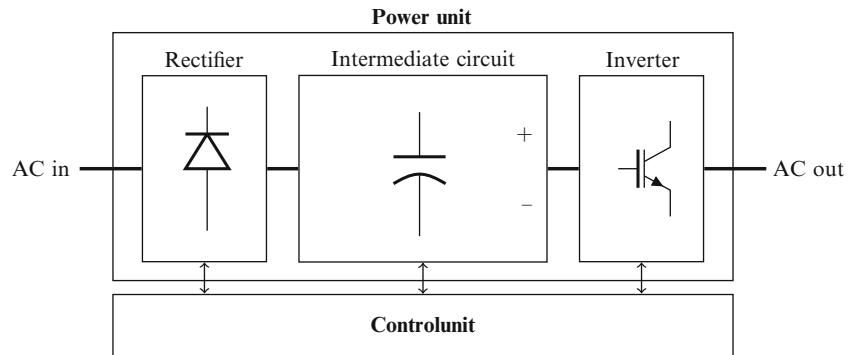
Figure 9.3: Components inside an AC drive
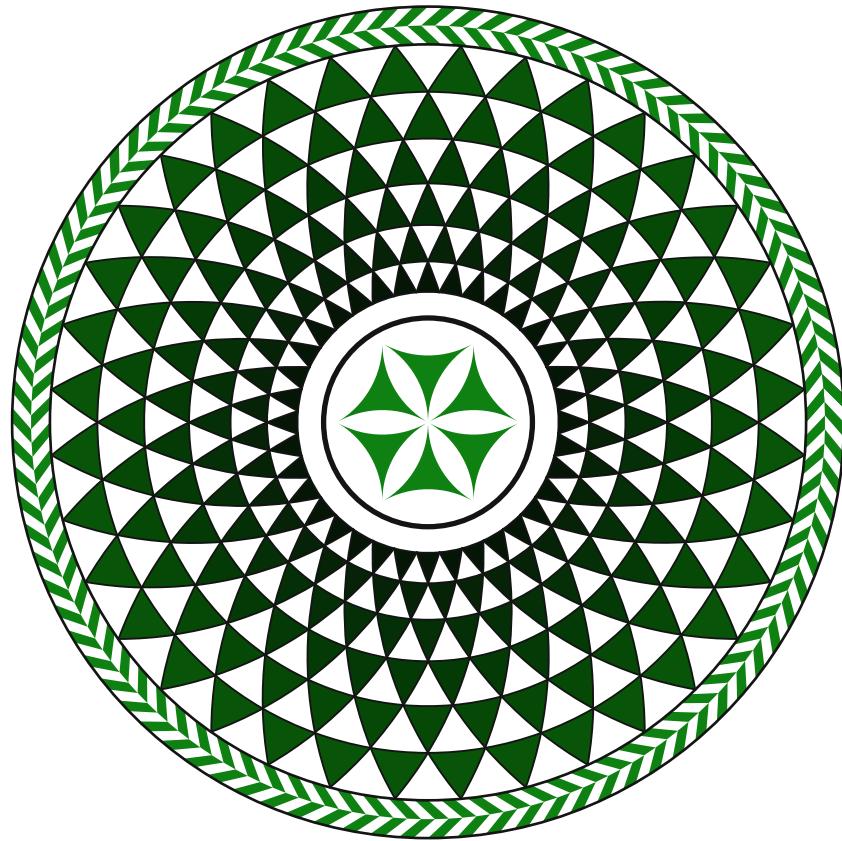
Figure 9.4: Mosaic from Pompeii
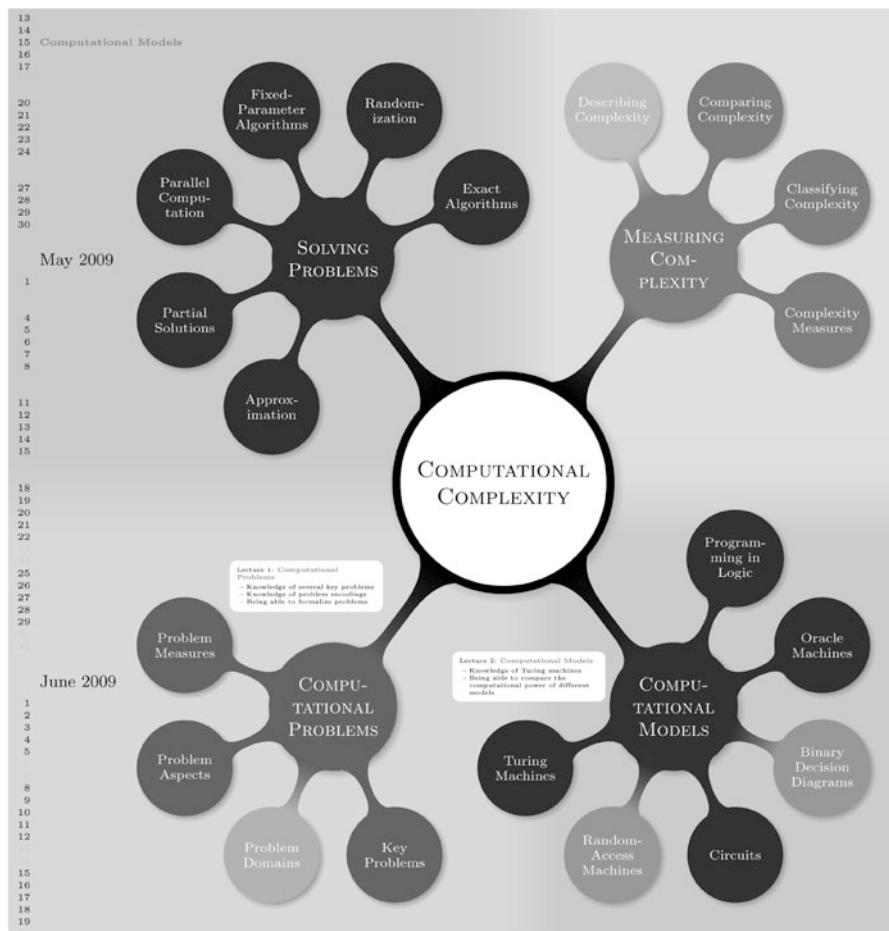
Figure 9.5: A course outline

A

Text symbol tables

A.1 Some European characters

Name	Type	Typeset	Type	Typeset
a-ring	\aa	å	\AA	Å
aesc	\ae	æ	\AE	Æ
ethel	\oe	œ	\OE	Œ
eszett	\ss	ß	\SS	SS
inverted question mark	?`	¿		
inverted exclamation mark	!`	¡		
slashed L	\l	ł	\L	Ł
slashed O	\o	ø	\O	Ø

A.2   Text accents

Name	Type	Typeset	Name	Type	Typeset
acute	\'{o}	ó	macron	\={o}	ō
breve	\u{o}	ŏ	overdot	\.{g}	ġ
caron/haček	\v{o}	ǒ	ring	\r{u}	ů
cedilla	\c{c}	ç	tie	\t{oo}	o͡o
circumflex	\^{o}	ô	tilde	\~{n}	ñ
dieresis/umlaut	\"{u}	ü	underdot	\d{m}	ṃ
double acute	\H{o}	ő	underbar	\b{o}	o̱
grave	\`{o}	ò			
dotless i	\i	ı	dotless j	\j	ȷ
	\'{\i}	í		\v{\j}	ǰ

A.3   Text font commands

A.3.1   Text font family commands

Command with Argument	Command Declaration	Switches to the font family
\textnormal{...}	{\normalfont ...}	document
\emph{...}	{\em ...}	*emphasis*
\textrm{...}	{\rmfamily ...}	roman
\textsf{...}	{\sffamily ...}	sans serif
\texttt{...}	{\ttfamily ...}	typewriter style
\textup{...}	{\upshape ...}	upright shape
\textit{...}	{\itshape ...}	*italic shape*
\textsl{...}	{\slshape ...}	*slanted shape*
\textsc{...}	{\scshape ...}	SMALL CAPITALS
\textbf{...}	{\bfseries ...}	**bold**
\textmd{...}	{\mdseries ...}	normal weight and width

A.3.2   Text font size changes

Command	AMS sample text
`\Tiny`	sample text
`\tiny`	sample text
`\SMALL` or `\scriptsize`	sample text
`\Small` or `\footnotesize`	sample text
`\small`	sample text
`\normalsize`	sample text
`\large`	sample text
`\Large`	sample text
`\LARGE`	sample text
`\huge`	sample text
`\Huge`	sample text

A.3.3   Special characters

Name	Type	Typeset
Ampersand	`\&`	&
Caret	`\^{}`	^
Dollar Sign	`\$`	$
Left Brace	`\{`	{
Right Brace	`\}`	}
Underscore (or Lowline)	`\_`	_
Octothorp	`\#`	#
Percent	`\%`	%
Tilde	`\~{}`	~

A.4   Additional text symbols

Name	Type	Typeset
ampersand	\&	&
asterisk bullet	\textasteriskcentered	∗
backslash	\textbackslash	\
bar (caesura)	\textbar	|
brace left	\{	{
brace right	\}	}
bullet	\textbullet	•
circled a	\textcircled{a}	ⓐ
circumflex	\textasciicircum	^
copyright	\copyright	©
dagger	\dag	†
double dagger (diesis)	\ddag	‡
dollar	\$	$
double quotation left	\textquotedblleft or ``	"
double quotation right	\textquotedblright or ''	"
em dash	\textemdash or ---	—
en dash	\textendash or --	–
exclamation down	\textexclamdown or !`	¡
greater than	\textgreater	>
less than	\textless	<
lowline	_	_
midpoint	\textperiodcentered	·
octothorp	\#	#
percent	\%	%
pilcrow (paragraph)	\P	¶
question down	\textquestiondown or ?`	¿
registered trademark	\textregistered	®
section	\S	§
single quote left	\textquoteleft or `	'
single quote right	\textquoteright or '	'
sterling	\pounds	£
superscript	^{a}	a
tilde	\textasciitilde	~
trademark	\texttrademark	TM
visible space	\textvisiblespace	␣

B

Math symbol tables

B.1   Hebrew and Greek letters

Hebrew letters

Type	Typeset
\aleph	א
\beth	ב
\daleth	ד
\gimel	ג

Greek letters

Lowercase

Type	Typeset	Type	Typeset	Type	Typeset
\alpha	α	\iota	ι	\sigma	σ
\beta	β	\kappa	κ	\tau	τ
\gamma	γ	\lambda	λ	\upsilon	υ
\delta	δ	\mu	μ	\phi	ϕ
\epsilon	ϵ	\nu	ν	\chi	χ
\zeta	ζ	\xi	ξ	\psi	ψ
\eta	η	\pi	π	\omega	ω
\theta	θ	\rho	ρ		
\varepsilon	ε	\varpi	ϖ	\varsigma	ς
\vartheta	ϑ	\varrho	ϱ	\varphi	φ
		\digamma	$\digamma$	\varkappa	$\varkappa$

Uppercase

Type	Typeset	Type	Typeset	Type	Typeset
\Gamma	Γ	\Xi	Ξ	\Phi	Φ
\Delta	Δ	\Pi	Π	\Psi	Ψ
\Theta	Θ	\Sigma	Σ	\Omega	Ω
\Lambda	Λ	\Upsilon	Υ		
\varGamma	$\varGamma$	\varXi	$\varXi$	\varPhi	$\varPhi$
\varDelta	$\varDelta$	\varPi	$\varPi$	\varPsi	$\varPsi$
\varTheta	$\varTheta$	\varSigma	$\varSigma$	\varOmega	$\varOmega$
\varLambda	$\varLambda$	\varUpsilon	$\varUpsilon$		

B.2   Binary relations

Type	Typeset	Type	Typeset
<	$<$	>	$>$
=	$=$	:	$:$
\in	$\in$	\ni or \owns	$\ni$
\leq or \le	$\leq$	\geq or \ge	$\geq$
\ll	$\ll$	\gg	$\gg$
\prec	$\prec$	\succ	$\succ$
\preceq	$\preceq$	\succeq	$\succeq$
\sim	$\sim$	\approx	$\approx$
\simeq	$\simeq$	\cong	$\cong$
\equiv	$\equiv$	\doteq	$\doteq$
\subset	$\subset$	\supset	$\supset$
\subseteq	$\subseteq$	\supseteq	$\supseteq$
\sqsubseteq	$\sqsubseteq$	\sqsupseteq	$\sqsupseteq$
\smile	$\smile$	\frown	$\frown$
\perp	$\perp$	\models	$\models$
\mid	$\mid$	\parallel	$\parallel$
\vdash	$\vdash$	\dashv	$\dashv$
\propto	$\propto$	\asymp	$\asymp$
\bowtie	$\bowtie$		
\sqsubset	$\sqsubset$	\sqsupset	$\sqsupset$
\Join	$\Join$		

Note the \colon command used in $f\colon x \to x^2$, typed as f \colon x \to x^2

Some of the symbols on this page and in the rest of this appendix require the latexsym and amssymb packages.

More binary relations

Type	Typeset	Type	Typeset
\leqq	$\leqq$	\geqq	$\geqq$
\leqslant	$\leqslant$	\geqslant	$\geqslant$
\eqslantless	$\eqslantless$	\eqslantgtr	$\eqslantgtr$
\lesssim	$\lesssim$	\gtrsim	$\gtrsim$
\lessapprox	$\lessapprox$	\gtrapprox	$\gtrapprox$
\approxeq	$\approxeq$		
\lessdot	$\lessdot$	\gtrdot	$\gtrdot$
\lll	$\lll$	\ggg	$\ggg$
\lessgtr	$\lessgtr$	\gtrless	$\gtrless$
\lesseqgtr	$\lesseqgtr$	\gtreqless	$\gtreqless$
\lesseqqgtr	$\lesseqqgtr$	\gtreqqless	$\gtreqqless$
\doteqdot	$\doteqdot$	\eqcirc	$\eqcirc$
\circeq	$\circeq$	\triangleq	$\triangleq$
\risingdotseq	$\risingdotseq$	\fallingdotseq	$\fallingdotseq$
\backsim	$\backsim$	\thicksim	$\thicksim$
\backsimeq	$\backsimeq$	\thickapprox	$\thickapprox$
\preccurlyeq	$\preccurlyeq$	\succcurlyeq	$\succcurlyeq$
\curlyeqprec	$\curlyeqprec$	\curlyeqsucc	$\curlyeqsucc$
\precsim	$\precsim$	\succsim	$\succsim$
\precapprox	$\precapprox$	\succapprox	$\succapprox$
\subseteqq	$\subseteqq$	\supseteqq	$\supseteqq$
\Subset	$\Subset$	\Supset	$\Supset$
\vartriangleleft	$\vartriangleleft$	\vartriangleright	$\vartriangleright$
\trianglelefteq	$\trianglelefteq$	\trianglerighteq	$\trianglerighteq$
\vDash	$\vDash$	\Vdash	$\Vdash$
\Vvdash	$\Vvdash$		
\smallsmile	$\smallsmile$	\smallfrown	$\smallfrown$
\shortmid	$\shortmid$	\shortparallel	$\shortparallel$
\bumpeq	$\bumpeq$	\Bumpeq	$\Bumpeq$
\between	$\between$	\pitchfork	$\pitchfork$
\varpropto	$\varpropto$	\backepsilon	$\backepsilon$
\blacktriangleleft	$\blacktriangleleft$	\blacktriangleright	$\blacktriangleright$
\therefore	$\therefore$	\because	$\because$

Negated binary relations

Type	Typeset	Type	Typeset
\neq or \ne	$\neq$	\notin	$\notin$
\nless	$\nless$	\ngtr	$\ngtr$
\nleq	$\nleq$	\ngeq	$\ngeq$
\nleqslant	$\nleqslant$	\ngeqslant	$\ngeqslant$
\nleqq	$\nleqq$	\ngeqq	$\ngeqq$
\lneq	$\lneq$	\gneq	$\gneq$
\lneqq	$\lneqq$	\gneqq	$\gneqq$
\lvertneqq	$\lvertneqq$	\gvertneqq	$\gvertneqq$
\lnsim	$\lnsim$	\gnsim	$\gnsim$
\lnapprox	$\lnapprox$	\gnapprox	$\gnapprox$
\nprec	$\nprec$	\nsucc	$\nsucc$
\npreceq	$\npreceq$	\nsucceq	$\nsucceq$
\precneqq	$\precneqq$	\succneqq	$\succneqq$
\precnsim	$\precnsim$	\succnsim	$\succnsim$
\precnapprox	$\precnapprox$	\succnapprox	$\succnapprox$
\nsim	$\nsim$	\ncong	$\ncong$
\nshortmid	$\nshortmid$	\nshortparallel	$\nshortparallel$
\nmid	$\nmid$	\nparallel	$\nparallel$
\nvdash	$\nvdash$	\nvDash	$\nvDash$
\nVdash	$\nVdash$	\nVDash	$\nVDash$
\ntriangleleft	$\ntriangleleft$	\ntriangleright	$\ntriangleright$
\ntrianglelefteq	$\ntrianglelefteq$	\ntrianglerighteq	$\ntrianglerighteq$
\nsubseteq	$\nsubseteq$	\nsupseteq	$\nsupseteq$
\nsubseteqq	$\nsubseteqq$	\nsupseteqq	$\nsupseteqq$
\subsetneq	$\subsetneq$	\supsetneq	$\supsetneq$
\varsubsetneq	$\varsubsetneq$	\varsupsetneq	$\varsupsetneq$
\subsetneqq	$\subsetneqq$	\supsetneqq	$\supsetneqq$
\varsubsetneqq	$\varsubsetneqq$	\varsupsetneqq	$\varsupsetneqq$

B.3   Binary operations

Type	Typeset	Type	Typeset
`+`	$+$	`-`	$-$
`\pm`	$\pm$	`\mp`	$\mp$
`\times`	$\times$	`\cdot`	$\cdot$
`\circ`	$\circ$	`\bigcirc`	$\bigcirc$
`\div`	$\div$	`\bmod`	mod
`\cap`	$\cap$	`\cup`	$\cup$
`\sqcap`	$\sqcap$	`\sqcup`	$\sqcup$
`\wedge` or `\land`	$\wedge$	`\vee` or `\lor`	$\vee$
`\triangleleft`	$\triangleleft$	`\triangleright`	$\triangleright$
`\bigtriangleup`	$\bigtriangleup$	`\bigtriangledown`	$\bigtriangledown$
`\oplus`	$\oplus$	`\ominus`	$\ominus$
`\otimes`	$\otimes$	`\oslash`	$\oslash$
`\odot`	$\odot$	`\bullet`	$\bullet$
`\dagger`	$\dagger$	`\ddagger`	$\ddagger$
`\setminus`	$\setminus$	`\smallsetminus`	$\smallsetminus$
`\wr`	$\wr$	`\amalg`	$\amalg$
`\ast`	$\ast$	`\star`	$\star$
`\diamond`	$\diamond$		
`\lhd`	$\lhd$	`\rhd`	$\rhd$
`\unlhd`	$\unlhd$	`\unrhd`	$\unrhd$
`\dotplus`	$\dotplus$	`\centerdot`	$\centerdot$
`\ltimes`	$\ltimes$	`\rtimes`	$\rtimes$
`\leftthreetimes`	$\leftthreetimes$	`\rightthreetimes`	$\rightthreetimes$
`\circleddash`	$\circleddash$	`\uplus`	$\uplus$
`\barwedge`	$\barwedge$	`\doublebarwedge`	$\doublebarwedge$
`\curlywedge`	$\curlywedge$	`\curlyvee`	$\curlyvee$
`\veebar`	$\veebar$	`\intercal`	$\intercal$
`\doublecap` or `\Cap`	$\Cap$	`\doublecup` or `\Cup`	$\Cup$
`\circledast`	$\circledast$	`\circledcirc`	$\circledcirc$
`\boxminus`	$\boxminus$	`\boxtimes`	$\boxtimes$
`\boxdot`	$\boxdot$	`\boxplus`	$\boxplus$
`\divideontimes`	$\divideontimes$	`\vartriangle`	$\vartriangle$
`\And`	$\&$		

B.4   Arrows

Type	Typeset	Type	Typeset
`\leftarrow`	←	`\rightarrow` or `\to`	→
`\longleftarrow`	⟵	`\longrightarrow`	⟶
`\Leftarrow`	⇐	`\Rightarrow`	⇒
`\Longleftarrow`	⟸	`\Longrightarrow`	⟹
`\leftrightarrow`	↔	`\longleftrightarrow`	⟷
`\Leftrightarrow`	⇔	`\Longleftrightarrow`	⟺
`\uparrow`	↑	`\downarrow`	↓
`\Uparrow`	⇑	`\Downarrow`	⇓
`\updownarrow`	↕	`\Updownarrow`	⇕
`\nearrow`	↗	`\searrow`	↘
`\swarrow`	↙	`\nwarrow`	↖
`\iff`	⟺	`\mapstochar`	∣
`\mapsto`	↦	`\longmapsto`	⟼
`\hookleftarrow`	↩	`\hookrightarrow`	↪
`\leftharpoonup`	↼	`\rightharpoonup`	⇀
`\leftharpoondown`	↽	`\rightharpoondown`	⇁
`\leadsto`	⇝		
`\leftleftarrows`	⇇	`\rightrightarrows`	⇉
`\leftrightarrows`	⇆	`\rightleftarrows`	⇄
`\Lleftarrow`	⇚	`\Rrightarrow`	⇛
`\twoheadleftarrow`	↞	`\twoheadrightarrow`	↠
`\leftarrowtail`	↢	`\rightarrowtail`	↣
`\looparrowleft`	↫	`\looparrowright`	↬
`\upuparrows`	⇈	`\downdownarrows`	⇊
`\upharpoonleft`	↿	`\upharpoonright`	↾
`\downharpoonleft`	⇃	`\downharpoonright`	⇂
`\leftrightsquigarrow`	↭	`\rightsquigarrow`	⇝
`\multimap`	⊸		
`\nleftarrow`	↚	`\nrightarrow`	↛
`\nLeftarrow`	⇍	`\nRightarrow`	⇏
`\nleftrightarrow`	↮	`\nLeftrightarrow`	⇎
`\dashleftarrow`	⇠	`\dashrightarrow`	⇢
`\curvearrowleft`	↶	`\curvearrowright`	↷
`\circlearrowleft`	↺	`\circlearrowright`	↻
`\leftrightharpoons`	⇋	`\rightleftharpoons`	⇌
`\Lsh`	↰	`\Rsh`	↱

B.5   Miscellaneous symbols

Type	Typeset	Type	Typeset
\hbar	ℏ	\ell	ℓ
\imath	ı	\jmath	ȷ
\wp	℘	\partial	∂
\Im	ℑ	\Re	ℜ
\infty	∞	\prime	′
\emptyset	∅	\varnothing	∅
\forall	∀	\exists	∃
\smallint	∫	\triangle	△
\top	⊤	\bot	⊥
\P	¶	\S	§
\dag	†	\ddag	‡
\flat	♭	\natural	♮
\sharp	♯	\angle	∠
\clubsuit	♣	\diamondsuit	♢
\heartsuit	♡	\spadesuit	♠
\surd	√	\nabla	∇
\pounds	£	\neg or \lnot	¬
\Box	□	\Diamond	◇
\mho	℧		
\hslash	ℏ	\complement	∁
\backprime	‵	\nexists	∄
\Bbbk	𝕜		
\diagup	╱	\diagdown	╲
\blacktriangle	▲	\blacktriangledown	▼
\triangledown	▽	\eth	ð
\square	□	\blacksquare	■
\lozenge	◇	\blacklozenge	◆
\measuredangle	∡	\sphericalangle	∢
\circledS	Ⓢ	\bigstar	★
\Finv	⅁	\Game	⅁

B.6 Delimiters

Name	Type	Typeset
left parenthesis	(	(
right parenthesis	)	)
left bracket	[or \lbrack	[
right bracket	] or \rbrack	]
left brace	\{ or \lbrace	{
right brace	\} or \rbrace	}
backslash	\backslash	\
forward slash	/	/
left angle bracket	\langle	⟨
right angle bracket	\rangle	⟩
vertical line	| or \vert	|
double vertical line	\\| or \Vert	‖
left floor	\lfloor	⌊
right floor	\rfloor	⌋
left ceiling	\lceil	⌈
right ceiling	\rceil	⌉
upward	\uparrow	↑
double upward	\Uparrow	⇑
downward	\downarrow	↓
double downward	\Downarrow	⇓
up-and-down	\updownarrow	↕
double up-and-down	\Updownarrow	⇕
upper-left corner	\ulcorner	⌜
upper-right corner	\urcorner	⌝
lower-left corner	\llcorner	⌞
lower-right corner	\lrcorner	⌟

B.7   Operators

"Pure" operators, with no limits

Type	Typeset	Type	Typeset	Type	Typeset	Type	Typeset
\arccos	arccos	\cot	cot	\hom	hom	\sin	sin
\arcsin	arcsin	\coth	coth	\ker	ker	\sinh	sinh
\arctan	arctan	\csc	csc	\lg	lg	\tan	tan
\arg	arg	\deg	deg	\ln	ln	\tanh	tanh
\cos	cos	\dim	dim	\log	log		
\cosh	cosh	\exp	exp	\sec	sec		

Operators with limits

Type	Typeset	Type	Typeset
\det	det	\limsup	lim sup
\gcd	gcd	\max	max
\inf	inf	\min	min
\lim	lim	\Pr	Pr
\liminf	lim inf	\sup	sup
\injlim	inj lim	\projlim	proj lim
\varliminf	$\underline{\lim}$	\varlimsup	$\overline{\lim}$
\varinjlim	$\varinjlim$	\varprojlim	$\varprojlim$

B.7.1 *Large operators*

Type	Inline	Displayed
\int_{a}^{b}	$\int_a^b$	$\int_a^b$
\oint_{a}^{b}	$\oint_a^b$	$\oint_a^b$
\iint_{a}^{b}	$\iint_a^b$	$\iint_a^b$
\iiint_{a}^{b}	$\iiint_a^b$	$\iiint_a^b$
\iiiiint_{a}^{b}	$\iiiint_a^b$	$\iiiint_a^b$
\idotsint_{a}^{b}	$\int\cdots\int_a^b$	$\int\cdots\int_a^b$
\prod_{i=1}^{n}	$\prod_{i=1}^n$	$\prod_{i=1}^n$
\coprod_{i=1}^{n}	$\coprod_{i=1}^n$	$\coprod_{i=1}^n$
\bigcap_{i=1}^{n}	$\bigcap_{i=1}^n$	$\bigcap_{i=1}^n$
\bigcup_{i=1}^{n}	$\bigcup_{i=1}^n$	$\bigcup_{i=1}^n$
\bigwedge_{i=1}^{n}	$\bigwedge_{i=1}^n$	$\bigwedge_{i=1}^n$
\bigvee_{i=1}^{n}	$\bigvee_{i=1}^n$	$\bigvee_{i=1}^n$
\bigsqcup_{i=1}^{n}	$\bigsqcup_{i=1}^n$	$\bigsqcup_{i=1}^n$
\biguplus_{i=1}^{n}	$\biguplus_{i=1}^n$	$\biguplus_{i=1}^n$
\bigotimes_{i=1}^{n}	$\bigotimes_{i=1}^n$	$\bigotimes_{i=1}^n$
\bigoplus_{i=1}^{n}	$\bigoplus_{i=1}^n$	$\bigoplus_{i=1}^n$
\bigodot_{i=1}^{n}	$\bigodot_{i=1}^n$	$\bigodot_{i=1}^n$
\sum_{i=1}^{n}	$\sum_{i=1}^n$	$\sum_{i=1}^n$

B.8   *Math accents and fonts*

Math accents

		amsxtra	
Type	Typeset	Type	Typeset
\acute{a}	$\acute{a}$		
\bar{a}	$\bar{a}$		
\breve{a}	$\breve{a}$	\spbreve	$\breve{}$
\check{a}	$\check{a}$	\spcheck	$\vee$
\dot{a}	$\dot{a}$	\spdot	$\cdot$
\ddot{a}	$\ddot{a}$	\spddot	$\cdot\cdot$
\dddot{a}	$\dddot{a}$	\spdddot	$\cdots$
\ddddot{a}	$\ddddot{a}$		
\grave{a}	$\grave{a}$		
\hat{a}	$\hat{a}$		
\widehat{a}	$\widehat{a}$	\sphat	$\widehat{}$
\mathring{a}	$\mathring{a}$		
\tilde{a}	$\tilde{a}$		
\widetilde{a}	$\widetilde{a}$	\sptilde	$\sim$
\vec{a}	$\vec{a}$		

Math fonts

Type	Typeset
LaTeX	
\mathbf{A}	$\mathbf{A}$
\mathcal{A}	$\mathcal{A}$
\mathit{A}	A
\mathnormal{A}	A
\mathrm{A}	A
\mathsf{A}	A
\mathtt{A}	$\mathtt{A}$
\boldsymbol{\alpha}	$\boldsymbol{\alpha}$
\mathbb{A}	$\mathbb{A}$
\mathfrak{A}	$\mathfrak{A}$
\mathscr{a}	$\mathscr{A}$

\mathscr requires the eucal package with the mathscr option

B.9 Math spacing commands

Name	Width	Short	Long
1 mu (math unit)	�ı	\mspace{1mu}	
thinspace	⊔	\,	\thinspace
medspace	⊔	\:	\medspace
thickspace	⊔	\;	\thickspace
interword space	⊔	\⊔	
1 em	⎿⏌		\quad
2 em	⎿__⏌		\qquad
Negative space			
1 mu	ı		\mspace{-1mu}
thinspace	⊔	\!	\negthinspace
medspace	⊔		\negmedspace
thickspace	⊔		\negthickspace

C

*L*A*T*E*X on the iPad*

A few years back, personal computing was desktop-centric. To update the operating system, for back up, and for many other tasks, you had to connect your smartphone and tablet with a computer. Tim Cook (Apple's CEO as I am writing this book) coined the term "Post PC revolution" to describe the trend that a tablet is no longer a younger brother of a PC, but an equal partner; in fact, for many users, it can be the only computer they will ever need.

But can you use it for your LaTeX documents? Isn't the iPad designed only for e-mail, to read news, and enjoy entertainment? Certainly. While it has a dual-core CPU, it has a quad-core graphics chip so viewing videos and complex Web pages is quick. The operating system is designed to make performing these basic tasks very easy and intuitive. iOS masks the complexities of the underlying computer.

Nevertheless, underneath this easy-to-use interface there is a Mac. Get a little familiar with the iPad as a computer, and you can work with your LaTeX documents pretty well.

There are good reasons why the iPad is the only tablet I'll discuss. Today, the iPad is clearly the dominant tablet of more than a hundred on the market and the iPad is the only tablet with a decent market share that is in an *ecosystem*: the iPad is just one device under iCloud along with the iPhone, the Mac desktops, and the Mac notebooks.

I work on a LaTeX document on my iMac, and when I am away from home, I continue my work on my MacBook Air or iPad; there is no interruption, all the devices are fully synchronized.

In Section C.1, we discuss the iPad file system, sandboxing, file transfers, printing, and text editing. In Section C.2, we briefly review why is it difficult to implement LaTeX on an iPad. We discuss where are the files to be LaTeXed and where the LaTeX process takes place in Section C.3. Finally, in Section C.4, we introduce two LaTeX implementations for the iPad: Texpad and TeX Writer.

This appendix is based on my articles in the Notices of the Amer. Math. Soc. **60** (2013), 332–334 and 434–439.

C.1   *The iPad as a computer*

To work on a document, Roth sits in front of his computer, in the complex folder hierarchy he finds `document.tex`, double clicks it to start the LaTeX implementation, edits it, typesets it. Then he prints `document.pdf`, proofreads it, and then he goes back to editing. . .

How do you work with these on an iPad? On the iPad, there is only a rectangular array of apps. No documents are visible. There may be folders containing more apps, but no folder in a folder. There are no Library folders, no Download folder. And no File menu containing the Print command!

I have `document.tex` on my desktop, but how do I transfer it to the iPad? I would plug in my thumb drive to facilitate the transfer, but there is no USB port.

In the Mac operating system, OS 10, there are always features missing. And we always hope that a future version will incorporate a solution. But this is different. These features are missing on purpose. Here is what Steve Jobs said about the file system: "You don't keep your music in the file system, that would be crazy. You keep it in this app that knows about music and knows how to find things in lots of different ways. Same with photos. . . And eventually, the file system management is just gonna be an app for pros, and consumers aren't gonna need to use it."

I will cover now the file system and sandboxing, file transfers, and printing for the iPad. Finally, I briefly introduce text editing.

C.1.1   *File system, sandboxing, and file transfers*

The iPad starts up displaying a rectangular array of icons and folders for apps; see Figure C.1. There are no icons for documents.

There is no familiar Desktop for documents and folders. No Applications folder. No multiple users. The screen is always occupied by a single window—creating difficulties with help screens that crowd out the screens they are supposed to help with. The file system, as we know it from desktop computers, is gone.
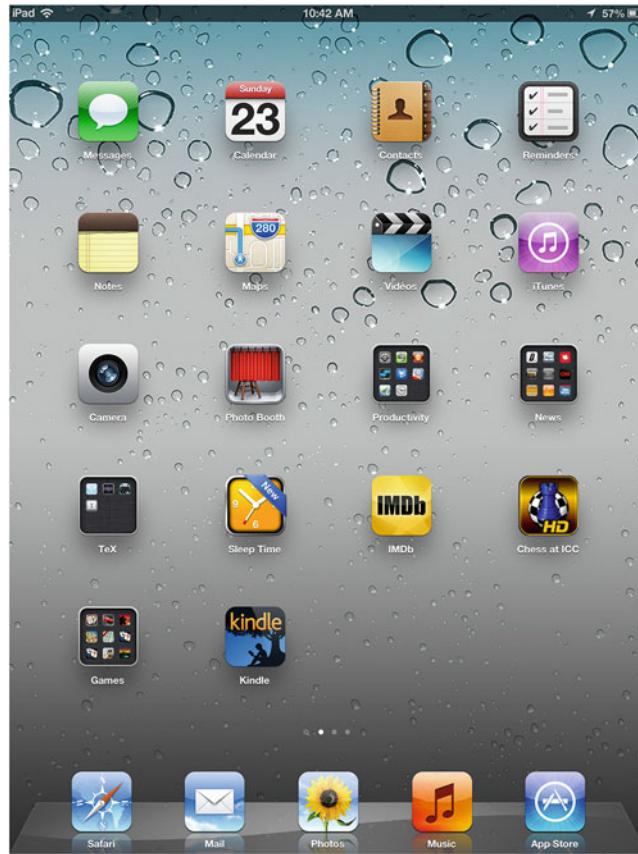
Figure C.1: A rectangular array of apps

In its place is an app-centric starting point. Touch the icon of an app and you are in business. When the app opens, you get access to the documents and settings of the app.

For security reasons, the apps are sandboxed, limiting an app's access to files, preferences, network resources, hardware, and so on. Ars Technica's John Siracusa described the goal of sandboxing as follows: "Running an application inside a sandbox is meant to minimize the damage that could be caused if that application is compromised by a piece of malware. A sandboxed application voluntarily surrenders the ability to do many things that a normal process run by the same user could do. For example, a normal application run by a user has the ability to delete every single file owned by that user. Obviously, a well-behaved application will not do this. But if an application becomes compromised, it can be coerced into doing something destructive."

Of course, the iPad is a computer, and it has a File System, we just do not see it. But it is important to visualize it. To help us along, we will use an app.

C.1.2   FileApp Pro

If you search the iPad's App Store for "file" apps, there are more than $1,000$ of them. Many of them could be used to help us understand the iPad file system. I choose FileApp Pro (by DigiDNA).
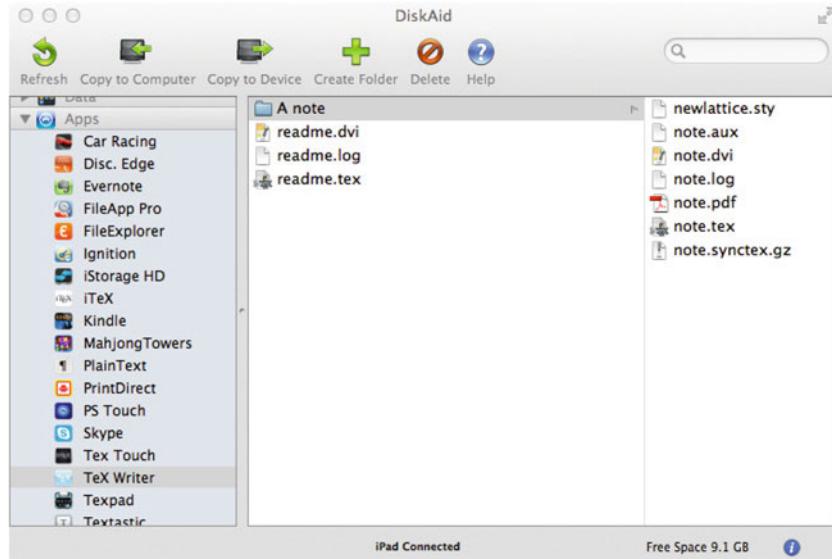
Figure C.2: DiskAid

To get started, plug the iPad into a desktop computer, download and start DiskAid on the computer; download and start FileApp (Pro) on the iPad. On the left panel of DiskAid, click on Apps, then on FileApp. The second pane now shows Imported Files, the right pane shows the files imported to the iPad; see Figure C.2. In FileApp, navigate to Imported Files. Anything you drag into the right pane of DiskAid is copied to FileApp's Imported Files. So much for file transfer. To see the file structure of the various iPad apps, click on their names. I choose TeX Writer.

File App Pro is a Swiss Army Knife of utilities. It can ZIP files, open ZIP files, create and edit text documents, record sound, and sing lullabies. Of course, for file transfers I should also mention the ubiquitous Dropbox. Download it for the iPad, sign in (as you did for your computer Dropbox); that's it.

C.1.3   Printing

When I first wanted to print from my iPhone, I realized that there is no print command. However, lots of apps would do the job. In fact, searching for "print" in the App Store, I discovered over 600 apps; many of them print, utilizing my desktop computer.

Typical of these apps is PrintDirect (EuroSmartz) and Printer Pro (Readdle Productivity). They can use any printer connected with your desktop computer. They wirelessly connect to your computer and print with its help.

If so many apps can help me out with printing, how come iOS does not? Read the comments about iOS printing; I was not the only one confused.

However, if the iPad is the poster child of the Post PC Revolution, its native printing solution cannot involve desktop computers. Apple introduced the appropriate technology; they named it AirPrint. The idea is simple: the iPad collaborates with the printer. Of course, for this you need a wireless printer that is AirPrint aware. Apple lists all the AirPrint aware printers:

http://support.apple.com/kb/ht4356+

as of this writing, about $1,000$. If you are lucky and have one of these printers, test it. Open an e-mail and touch the Action icon (here it is the Reply icon); this offers you the options: Reply, Forward, and Print. Touch Print. Printer Options appears, and you can choose how many copies and on which printer. (Lots of apps provide more choices, such as page range.) Choose the printer and print.

For a second test, open a Web page in Safari. There is only one difference: the action icon is a curved arrow in a rectangle.

As a third test, open the Drudge Report. It has the familiar Action icon; we are in business. Finally, open the Politico app, read the news and look for an action icon. There is none. So to use AirPrint, you need an AirPrint aware printer and an AirPrint aware app! For the time being, these are limiting restrictions.

C.1.4 Text editors

Many of us edit LaTeX documents in text editors more sophisticated than the text editor that comes with the LaTeX implementation. Some thoughts on iPad text editors.

First, writing about apps is like shooting at a moving target. While I was writing about an app, it went through four versions. Adding features, removing bugs.

Second, there are so many text editors, well over 200... Take a look at the table at

http://brettterpstra.com/ios-text-editors/

This table is a 103×31 matrix (as I'm writing this), each row representing a text editor, each column representing a feature (such as Search and Replace). The entries are Yes or No. Hovering over the name of a text editor, you get a listing of additional features and the App Store information.

Third, keeping the iPad horizontal, the keyboard gobbles up too much real estate. Keeping it vertical, the keyboard is less intrusive, but the keys are smaller. If you want to do serious work on the iPad, buy a keyboard.

Fourth, the iOS's touch text editing is nice, but it lacks a feature crucial for text editing: moving the cursor a character ahead or back. (Of course, keyboards have cursor keys!) Text editors offer a variety of solutions, for instance, finger swiping.

I will discuss briefly a very sophisticated text editor: Textastics. If you want Syntax Highlighting, Search and Replace, and Text Expander, this a good choice. In Figure C.3, you see me editing a document.

You can see the extra keyboard row and the cursor navigation wheel (which appears with a two finger tap—finger swipe also moves the cursor). It comes with an excellent user manual. (Textastics can also perform a number of non-editing tasks, such as zipping and unzipping files.) Textastics has a Mac version. And if you spend time shaping it to your liking, then you would like the same tamed editor for all your work.

C.2   *Sandboxing and GPL*

To implement LATEX on an iPad, two major—man-made—obstacles have to be overcome: Sandboxing and the GPL license.

We discussed sandboxing in Section C.1.1. Does it impact LATEX implementations? You bet. For instance: The LATEX implementation Texpad on the Mac is given a single LATEX root file; it then reads through the LATEX source, gets all the included files, and presents you with an outline of your project. Sandboxing would not allow this. The handling of the auxiliary files also poses a problem. Of course, these problems can be overcome by ingenious programmers.

Richard Stallman, of Emacs fame, started the GNU operating system in 1983. Soon after, he started a nonprofit corporation called the Free Software Foundation. Stallman wrote, with the assistance of some law professors, the General Public License (GPL)—the most widely used free software license—released in 1989. Version 3 is dated June 29, 2007, the day the iPhone was released. Many software developers use GPL to ensure the free distribution of their software (source code and executable) under reasonable terms. Some software developers seem not to be aware of the fact that GPL licensed software cannot be used in an app created for the iPad. Two well known developers told to me that they use GPL because their peers do. Both would like to get out of it but do not know how. How ironic: the license that was supposed to allow you to spread your free software to wherever it is needed, now stops you from having it used on the fastest growing platform of all time.

C.3   *Files and typesetting*

C.3.1   *Getting the files*

The LATEX files, of course, can always be composed in the app. But typically you already have them. You can obtain your existing files in two ways:

1. Using iTunes. To transfer files—one at a time—to your app from your computer using iTunes, connect your iPad to your computer and start iTunes by double clicking on its icon. Under Devices, we selected the iPad from the left side of the iTunes window; see Figure C.4. At the top of the iTunes window, next to Summary
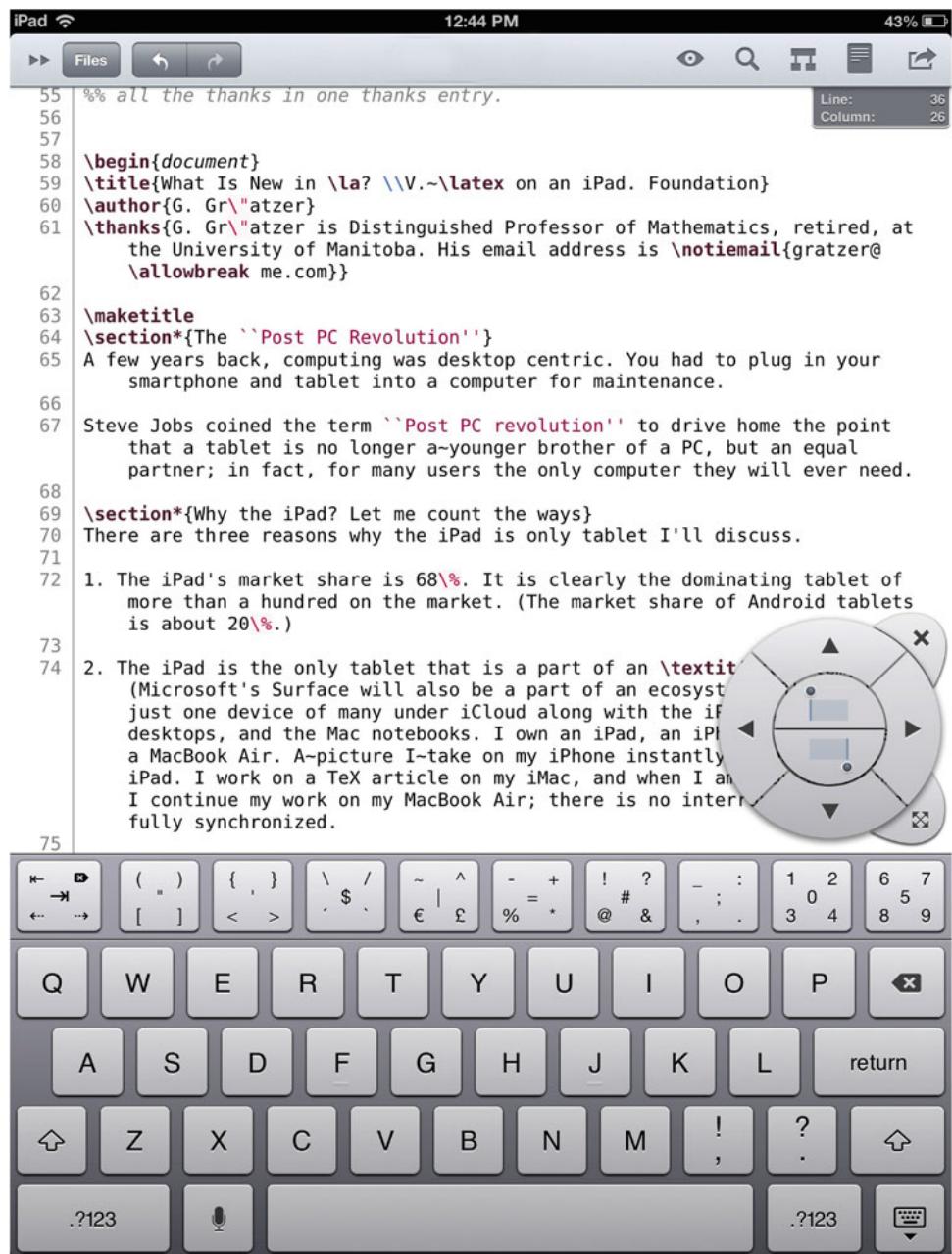
Figure C.3: Editing with Textastics
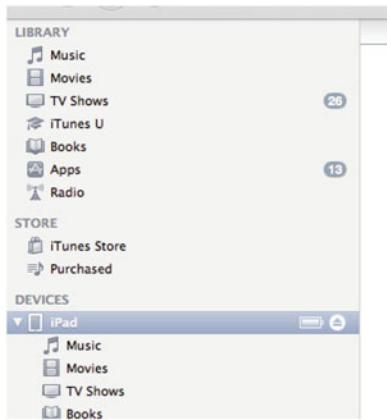
Figure C.4: Under Devices, we selected the iPad

Figure C.5: Choose Apps

and Info, select Apps; see Figure C.5. The lower part of the window now has File Sharing; see Figure C.6. On the left, you see a listing of the apps available for file transfer. Select the app; the files already in the app are then listed in the right pane. Click on the add button and a file browser appears. Choose the file you want to transfer.

   2. Via Dropbox. I assume that you have the ubiquitous Dropbox (the application that keeps your files safe and up-to-date across multiple devices and platforms). For an introduction, go to dropbox.com. In the app, you sign in to Dropbox. Now the app can see the contents of your Dropbox, or some part of it (at the Dropbox server) as long as you have an Internet connection.
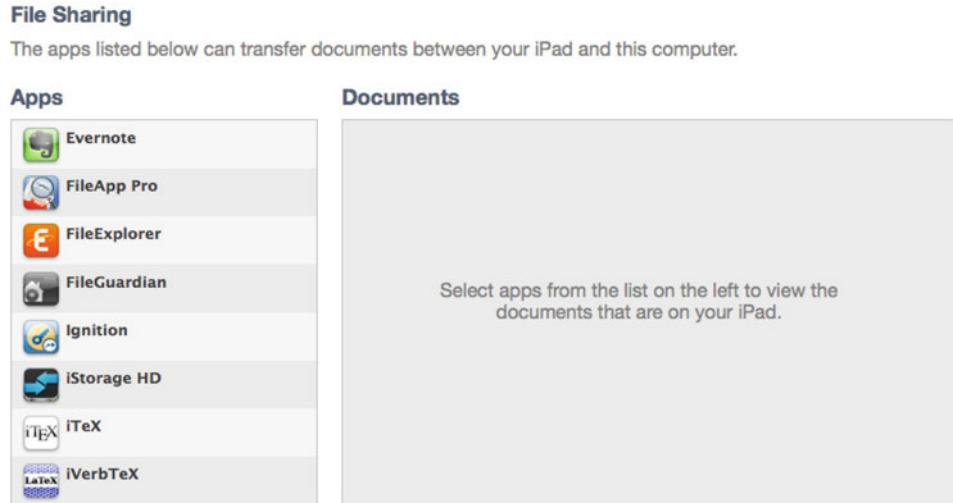
Figure C.6: Select app

C.3.2 Typesetting

The app can typeset the LaTeX file in the following ways:

A. On your iPad. This is the "Post PC revolution" option: the app places a LaTeX distribution on the iPad and you typeset with it. No computer or Internet connection is required. However, a complete LaTeX distribution is about 4 GB! No app can be this big. So you only get a minimal LaTeX distribution.

B. On your computer via Dropbox. This is the most powerful option. You have all the packages and fonts on your computer available to you. An app (such as AutomaTeX by Jonathan Weisberg) monitors if there is any change in the LaTeX file in Dropbox. If there is, the file is retypeset and the pdf is made available to you via the Dropbox.

C. In the cloud. This option provides you with a remote server, the Cloud; you connect to it with Wi-Fi. The server has a full LaTeX implementation, so you miss only the special fonts. And, of course, you must have Wi-Fi to use it. So you cannot polish up your lecture on the airplane on the way to a meeting.

Originally, the LaTeX output was a dvi file. These days, utilizing pdftex (under GPL license) by Hàn Thế Thành, the output is pdf. Since developers could not use GPL-d code, the output was dvi. These days, even on the iPad, pdf rules. In a more perfect world, these talented developers would not have to spend so much time reinventing GPL-d wheels.

```
\author[G.\ Cz\'edli]{G\'abor Cz\'edli}
\email{czedli@math.u-szeged.hu}
\urladdr{http://www.math.u-szeged.hu/$\sim$czedli/}
\address{University of Szeged\\ Bolyai Institute\\Szeged,
Aradi v\'ertan\'uk tere 1\\ Hungary 6720}
\author[G.\ Gr\"atzer]{George Gr\"atzer}
\email{gratzer@me.com}
\urladdr{http://server.math.umanitoba.ca/homepages/gratzer/}
\address{Department of Mathemat
MB R3T 2N2\\Canada}
```

Figure C.7: Editing with soft keyboard

```
\author[G.\ Gr\"atzer]{George Gr\"atzer}
\email{gratzer@me.com}
\urladdr{http://server.math.umanitoba.ca/homepages/gratzer/}
\address{Department of Mathematics\\University of Manitoba\\Winnipeg,
MB R3T 2N2\\Canada}

\thanks{This research was supported by the NFSR of Hungary (OTKA),
grant no. K77432}

\subjclass[2000]{Primary: 06B10, Secondary: 08A30}
\keywords{Lattice, tolerance, congruence}

\date{\today}

\begin{abstract}
We prove that a tolerance relation of a lattice
is a homomorphic image of a cong
```

Figure C.8: Editing with Bluetooth keyboard

C.3.3   *Keyboard or not to keyboard...*

In Figure C.7, you see editing with the iPad's soft keyboard (notice the extra row of LaTeX keys added by the LaTeX implementation, Texpad) and in Figure C.8, editing with a Bluetooth keyboard (notice that the extra row of LaTeX keys of Texpad is still present).

C.4   Two L*A*T*E*X implementations for the iPad

We now discuss two L*A*T*E*X implementations that typeset on the iPad.

C.4.1   Texpad

Files: Via Dropbox. **Typesetting:** On your iPad, on your computer via Dropbox, in the cloud.
Documentation: Excellent and detailed on the iPad interface. It is available as a help file and also at

<http://texpadapp.com/app-help-files/ios/help.html>

A. On your iPad. This is the "Post PC revolution" option: the app places a L*A*T*E*X distribution on the iPad and you typeset with it. No computer or Internet connection is required. However, a complete L*A*T*E*X distribution is about 4 GB! No app can be this big. So you only get a small L*A*T*E*X distribution.

B. On your computer via Dropbox. This is the most powerful option. You have all the packages and fonts on your computer available to you. An app (such as AutomaTeX by Jonathan Weisberg) monitors if there is any change in the L*A*T*E*X file in Dropbox. If there is, the file is retypeset and the pdf is made available to you via the Dropbox.

C. In the cloud. This option provides you with a remote server, the Cloud; you connect to it with Wi-Fi. The server has a full L*A*T*E*X implementation, so you miss only the special fonts. And, of course, you must have Wi-Fi to use it. So you cannot polish up your lecture on the airplane on the way to a meeting.

Texpad is a L*A*T*E*X implementation for the Mac and for the iPad. It has some interesting features, including:

- Autocompletion of all common commands and autofilling \cite-s and \ref-s.

- Replacement of the L*A*T*E*X console with a list of errors and warnings linked to the source.

- Global search, outline view, and syntax highlight.

Step 1. To get started with Texpad, go to the iPad App Store and install Texpad.
Sign up for Dropbox with the same e-mail address and password as for your computer's Dropbox.
Step 2. Now open Texpad. Figure C.9 shows Texpad at the first startup.
The Help button gets the help file.
Step 3. Touch Off to turn Dropbox On. (If you have Dropbox installed and connected, it's even simpler, you just have to Allow the connection.) Your File Storage now gives two options: iPad and Dropbox; see Figure C.10. It is important to understand that your L*A*T*E*X files will live in the Dropbox (in the Cloud, at the Dropbox server) or locally on your iPad.
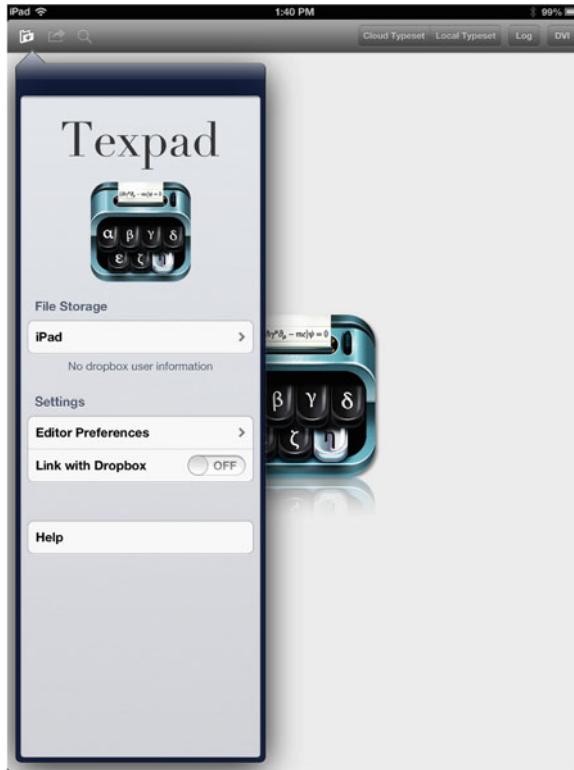
Figure C.9: Texpad first start up

Figure C.10: Expanded File Storage

Step 4. The Dropbox files are now available to you by touching Dropbox under File Storage, see Figure C.10.

- First, create a folder for the LATEX files to be transferred. Navigate to iPad file storage. Touch the + in the bottom right, and choose Folder. Name the folder.

- Second, navigate to the Dropbox file system view and to the folder containing the file you want to copy. Touch Edit. Select the file to transfer. At the bottom center, touch Copy. Navigate to the folder into which you want to copy the file and touch Copy.

Step 5. Typesetting will take place either on the iPad or in the Cloud. Go to the folder of a LATEX file, touch the file (on the iPad or in the Dropbox), and typeset it on the iPad (touch Local Typeset) or in the Cloud, that is, at Valletta's server (touch Cloud Typeset).

Step 6. Try to visualize what is happening.

- If you typeset on the iPad and the file is on the iPad, it just typesets locally; that is it.

- If you typeset on the iPad and the file is in Dropbox, the file is transferred to the iPad, typeset, and the resulting pdf is sent back to the Dropbox; nothing is kept at the iPad.

- If you typeset in the Cloud and the file is in Dropbox, the file is transferred to the Cloud, typeset, and the resulting pdf is sent back to the Dropbox; nothing is kept in the Cloud.

- If you typeset in the Cloud and the file is on the iPad, the file is transferred to the Cloud, typeset, and the resulting pdf is sent back to the iPad.

Step 7. Once you touch a LATEX file, you are ready to edit it. Cursor control is very important. You do it with a two finger swipe. Of course, this is not so important if you use a Bluetooth keyboard; it has cursor keys. But two finger swipe is faster!

Step 8. You edited and typeset your LATEX file. You want to get to another file. Touch the organize button (the folder icon on the upper left). You get the Organizer window; see Figure C.11. Touch the button in the upper left of the window, you get back to Dropbox, eventually, to the expanded File Storage of Figure 7.

These eight steps should be enough to get you started. Read the Help file for some more information.

C.4.2   TeX Writer

Files: Via Dropbox. **Typesetting:** On your iPad.
Documentation: The file `readme.pdf` is no quick start, but it is useful for understanding how TeX Writer works and how to customize it. TeX Writer was the first to typeset on the iPad. It could only typeset TeX files. Now it has LATEX and the AMS packages on board.

Step 1. When you start up TeX Writer, first link to Dropbox. In TeX Writer, you get a display showing the source file `readme.tex`; see Figure C.12. Pressing the More icon (right pointing arrow), you get more icons, to read the pdf version or Air Printing `readme.pdf`. On the left is the Organize icon; touching it, you get a file listing: `readme.tex` and `readme.pdf`. At the bottom is New File; touch it to compose one.
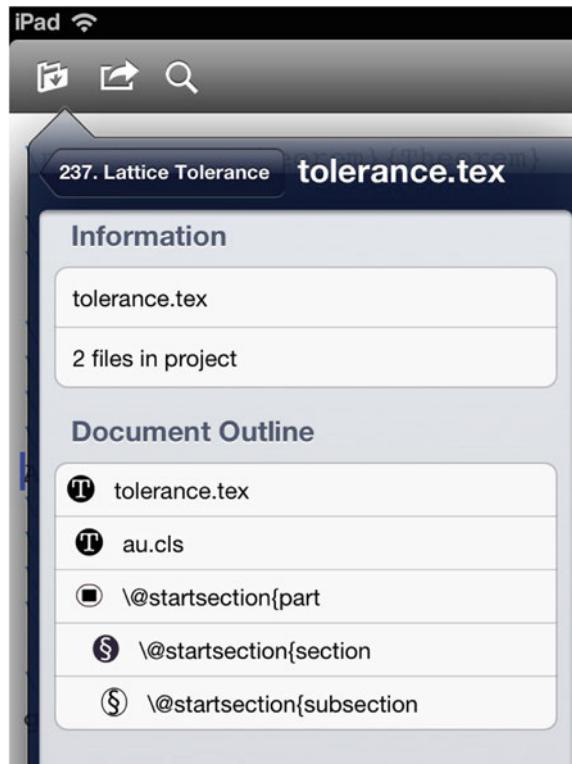
Figure C.11: Organizer window

Step 2. So you are perplexed about what to do next, you ran out of icons. You have to know that TeX Writer accesses the Dropbox in a special way. When you connect to Dropbox from TeX Writer, it creates a new folder `App` in Dropbox. In the folder `App` it creates the subfolder `TeX Writer`. In this subfolder you find `readme.tex`. Anything you put in the `TeX Writer` subfolder is visible in the file listing window on the iPad; anything not in this subfolder is not visible to TeX Writer.

Step 3. TeX Writer gets your files from this subfolder in Dropbox. Place a folder in there with the files of your current project. These will be available to you on your iPad. Moreover, these files are fully synchronized, so the editing changes you make on your iPad show up in Dropbox.

Step 4. LATEXing, you spend most of your time editing. TeX Writer's editor has some interesting features. Excellent cursor control. Touch `begin{}` type in the name of the environment, and the environment is placed in your document; undo, redo, search, and so on.

When typing, you retain the editing functions you get at the start, and in addition, you get an extra row of LaTeX specific keys. You do not get them with a Bluetooth keyboard; however, the keyboard can have many of these keys you need for typing LaTeX. Nice feature: the Log viewer links to error lines.
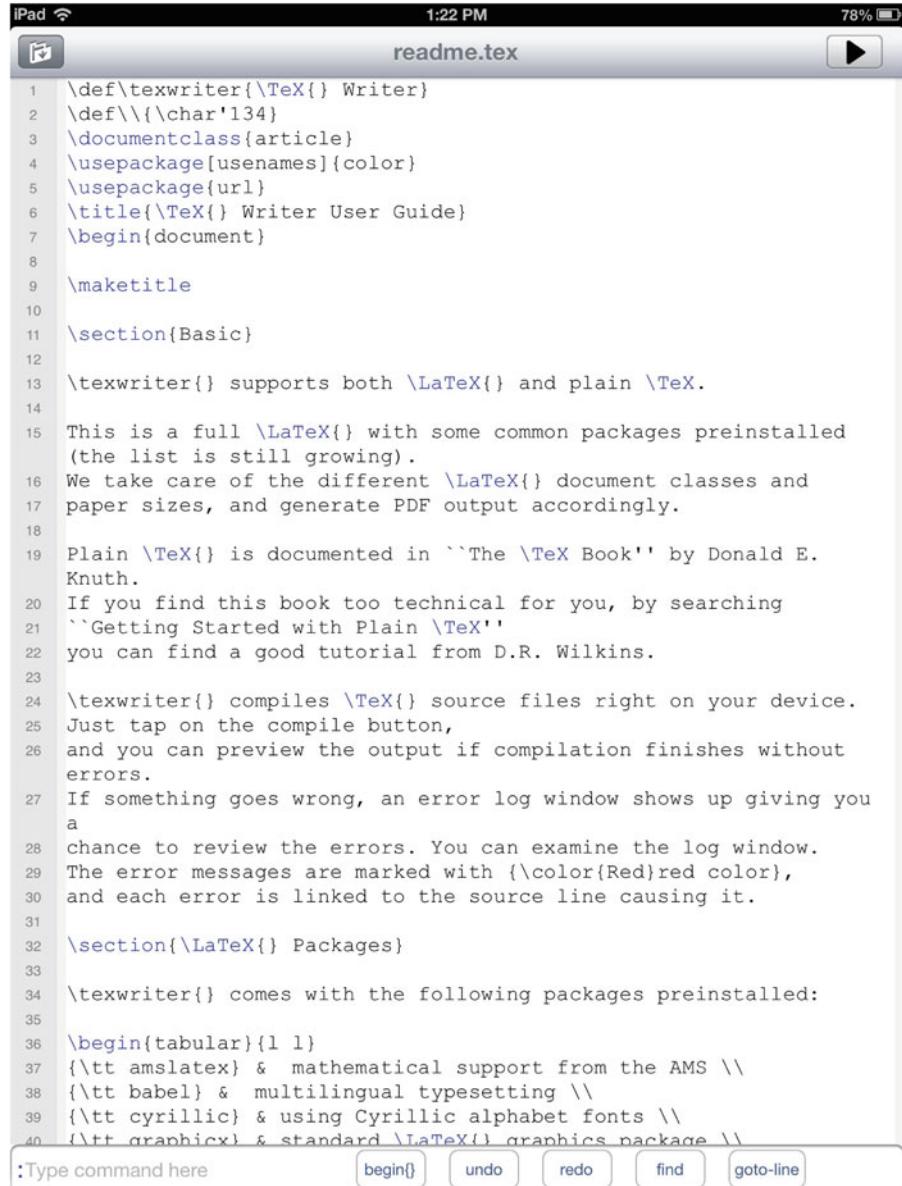
Figure C.12: TeX Writer startup

C.5   *Conclusion*

Jason Snell was interviewing Craig Federighi, Apple senior vice president of software engineering (and two more executives of Apple), for MacWorld. Snell writes:

"When I walked into Apple's offices for my conversation with the three executives, they noticed that I had brought a phone, a tablet, and a laptop, and had ultimately selected my MacBook Air as my tool of choice for the interview.

'You had a bunch of tools,' Federighi said, pointing at my bag. 'And you pulled out the one that felt right for the job that you were doing. It wasn't because it had more computing power. You pulled it out because it was the most natural device to accomplish a task.' "

I'm not suggesting that you write all your document on an iPad. I do suggest, however, that you can LaTeX with ease, say on a trip, correcting a document or adding a slide to your presentation. Use your iPad to LaTeX when appropriate.

LaTeXing on an iPad requires some compromises, for instance, you cannot use nonstandard fonts. Nevertheless, when not at your desk, the iPad will be nearly as functional as your MacBook Air, and it is so much easier to carry around...

Practical Finder