The Jolly Writer

The Jolly Writer
© 2020, Joris van der Hoeven

Dépot légal : août 2020
ISBN : 978-2-9572100-0-8
Première édition

Achevé d'imprimer en août 2020 par WirmachenDruck
Mühlbachstraße 7, 71522 Backnang, Allemagne
Imprimé en Allemagne

Scypress
6, rue de Versailles
78460 Chevreuse

The Jolly Writer

Joris van der Hoeven

Table of contents

Preface

The ultimate aim of a typesetter is to compose books in such a perfect manner that nobody will notice it. Nothing is more dishonorable than to make a reader stumble over an ill-placed comma. What the typesetter wants is to bury you cozily in your armchair, with your complete focus on the book, and in a spell not to be broken by trivial typographic details.

If you are a writer instead of a reader, then it takes far more than an armchair to put you in the right creative state of mind. The ideal text editor should help you to maintain this state for as long as possible. You don't want to be torn out of trance by a misguided comma key that suddenly starts inserting apostrophes. The ideal writing tool should behave as your first art pen from school: it should unleash your desire to write, make you feel one with the tool, and occasionally surprise you with the beauty of your own words.

Back in 1996, when I was writing my PhD in computer science, none of the existing text editors were even close to this ideal. General purpose editors such as MICROSOFT WORD produced documents of poor quality and made it difficult to type mathematical formulas. In order to compose professional-looking scientific documents, the main alternative was to use (LA)TEX. Although this solution was nice from the reader's perspective, it forced writers to encode their prose in a technical pseudo-language. (LA)TEX then relied on an akward "compilation" process in order to transform this pseudo-code into a printable and human-readable document.

After my PhD, this unsatisfactory state of the art led me to start the development of GNU TEXMACS, a free office suite for scientists. Before anything else, TEXMACS allows you to create beautiful scientific documents with special types of content, such as mathematical formulas or technical pictures. TEXMACS also provides interfaces for various external systems for symbolic and numeric computations. Recent versions further include a laptop presentation facility, versioning tools, integrated documentation, etc. In the areas of science and education, the ultimate aim is to provide a complete suite for the most frequent tasks on a computer.

One distinctive feature of TEXMACS is that it does not compromise on quality. First of all, the final documents have a professional typesetting quality, similar or even superior to what is achieved by (LA)TEX. Yet, user-friendliness has not been sacrificed, since the system also provides an intuitive wysiwyg (what-you-see-is-what-you-get) graphical interface. Finally, TEXMACS favors the composition of so-called "structured" documents, in contrast to existing wysiwyg interfaces of general purpose office suites, which are mainly "presentation oriented".

The name “GNU TeXmacs” is explained by the facts that TeXmacs is part of the GNU project and that some initial inspiration was drawn from the (LA)TeX systems [36, 38] and GNU EMACS [53]. However, it has become clear over time that this choice of name was one of the biggest mistakes of the project. Indeed, the name incorrectly suggests that TeXmacs is some kind of interface to (LA)TeX. So let me stress once and for all: the current TeXmacs system is completely independent from (LA)TeX. Of course, TeXmacs does provide converters between its native format and LaTeX.

As a free software, the development of TeXmacs has benefitted from the help of a wide community of contributors around the globe. Particular thanks go to the following main co-developers throughout the years: David ALLOUCHE, Miguel DE BENITO DELGADO, Darcy CHEN, Andrey GROZIN, Massimiliano GUBINELLI, Philippe JOYEZ, Grégoire LECERF, Henri LESOURD, François POULAIN, and Denis RAUX. Further thanks go to the many other contributors and supporters; see `http://www.texmacs.org/tmweb/about/authors.en.html` for a more extensive list. I am indebted to Basile AUDOLY and Kostas OIKONOMOU for their careful proofreading.

I also wish to express my gratitude to those who have provided financial support to the TeXmacs project: CNRS, CRI-TECH de Haute-Savoie, DIGITEO, Rennes MÉTROPÔLE, SPRINGER-VERLAG, INRIA, Dan GRAYSON, and Christoph BENZMUELLER.

TeXmacs is built on top of a lot of other software. The QT, GUILE, FREETYPE, and HUMMUS libraries are particularly essential [47, 33, 63, 34]. Recent versions integrate an increasing amount of artwork, such as the TeX Gyre fonts [32] (the main font of this book is PAGELLA), various “Subtle Patterns” [61], more fonts from DAFONT [9], and further freely available pictures from WIKIMEDIA [65].

Last but not least, I wish to thank SYLVIE, JUDITH, and NIELS for their support and patience.

Chapter 1
Introduction

1.1 How to read this book

The fact that you are reading these lines makes it is quite likely that you are already a convinced TeXmacs user. Maybe you actually bought this book in order to support our development efforts rather than to actually read it. Indeed, we tried to make TeXmacs so user-friendly that you will naturally learn the software while using it. The system also ships with a lot of integrated documentation that is most easily searched and consulted from within TeXmacs itself.

So why read this book? If you are a TeXmacs user or plan to become one, then you probably spend a considerable amount of your time on writing documents, doing computations, or making presentations from your laptop. In comparison, reading parts of this book will be a minor investment that might help you get the most out of TeXmacs.

On top of being an up-to-date manual for TeXmacs 2.1, this book indeed contains a substantial amount of information that is not so obvious to acquire just by using the software. For example, you can learn some of the basics of professional typesetting; typing kilometers of text will not miraculously instill such knowledge into your mind. Similarly, although every single feature of TeXmacs is rather self-explanatory, interesting results can be obtained through less obvious combinations of them. In this book, we present several tips and tricks of this kind.

One of the original motivations for developing TeXmacs was that traditional office suites such as Microsoft Office [42] and LibreOffice [13] offered poor support for scientific users. For instance, the typesetting quality of mathematical formulas was low in comparison to (LA)TeX [36, 38] and visual formatting was favored over structural markup (see section 1.3). For this reason, TeXmacs is particularly well suited for scientists, teachers, and students. But other types of users may also appreciate the combination of high typesetting quality, a wysiwyg front-end (what you see is what you get), and the possibility to write structured documents [28, 21].

As a non-native English speaker, the one thing I cannot teach you is how to write English. Pinker's *The Sense of Style* [44] is one excellent book that I can recommend to any fellow scientist who wants to become a better writer. Of course, you have a wealth of other options, such as Strunk and White's *The Ele-*

ments of Style [55] or Fowler's *Dictionary to Modern English Usage* [14]. Besides these classics, a few more recent style guides are Thomas and Turner's *Clear and Simple as the Truth: Writing Classic Prose* [60] and Williams's *Toward clarity and grace* [66]. *The Cambridge Grammar of the English Language* [31] is one of the bibles on grammar, whereas *Merriam-Webster's Dictionary of English Usage* is a dictionary that elevates words beyond the ordinary.

This book is organized as follows. The remainder of the present chapter is devoted to a quick introduction to some of the main features of $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ and its design philosophy. The next chapter discusses the the graphical user interface. These two chapters introduce the most important concepts and terminology, so we recommend that you read them first. The other chapters are fairly independent and organized by increasing order of difficulty.

Let us mention a few conventions that are followed throughout this book. We use a sans serif font for menu entries: “you may create a new file using File ▸ New”. Sometimes, clicking on a menu button will open a new popup window in which you have to click on yet another series of buttons. The menu notation naturally extends to this case, as in Edit ▸ Preferences ▸ Convert ▸ Pdf.

We use boxes such as ⌘⇧x and - > for keyboard shortcuts. The first example ⌘⇧x corresponds to simultaneously pressing the modifier keys ⌘ (meta) and ⇧ (shift) together with the letter x. In the second example - >, you first have to press and release the - key and next press and release the > key. For more information, we refer to section 2.3.1.

Notice that the default behavior of the menus and keyboard depends on your operating system. Indeed, $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ attempts to follow the same conventions as the other programs on your computer, and these conventions may vary with your operating system. The default behavior can be overridden by selecting another “look and feel” in the user preferences (Edit ▸ Preferences ▸ General). The keyboard shortcuts in this book hold for the EMACS look and feel on MAC computers. That said, many of them work for all look and feels, up to minor prefix adjustments. See section 2.3.6 for more details.

1.2 Making science beautiful

Modern word processors have made it easy for just about anyone to create nice-looking documents. Yet, there exists a difference between “nice” and “beautiful”. A few decades ago, typesetting books was the job of professionals. Whereas most of the typesetting can nowadays be carried out automatically, some design decisions still have to be made by humans. So let us step back and ask ourselves what makes a document attractive.

Typography is a matter of technical craftsmanship. Although we might compare it with architecture or the building of a bridge, typography does not involve emotions, as do arts like poetry, painting, or literature.

One fundamental idea about the esthetics of technology is that beauty is strongly correlated with adequacy to the purpose. A skillfully conceived and presented book usually provokes a quiet feeling of harmony, whereas clumsy formatting or an abundance of exuberant fonts will distract and thereby irritate the reader. In a sense, the better the typesetting quality, the less you will notice it. An advertiser, on the contrary, will use flashy colors and bold fonts to attract your attention. This may convince you to buy washing powder, but it does not make you better at differential geometry.

How to put this into practice? When starting a new document, you will first have to decide on the global layout, such as the page size, the extent of the actual text, the number of columns, the main fonts, etc. Here are a few useful rules:

- Do *not* make your paragraphs too wide, since wide paragraphs make it harder for the eye to find the next line, and thereby cause fatigue for the reader.

- Consistently stick to the same layout throughout your document. Use a different layout only if there is a special reason for it.

- In particular, keep the number of fonts to a minimum. Use alternative fonts for specific purposes only.

- Use fonts and a general layout that are well suited to the medium of publication and to the intended purpose. For example, sans serif fonts are more readable on a computer screen (so you might prefer them for web pages and laptop presentations) but may look unprofessional on paper.

To ease the choice of a general layout, TEXMACS offers *document styles* for various purposes: you may select Document ▸ Style ▸ Article for writing an article, Document ▸ Style ▸ Beamer for creating a laptop presentation, and so on. TEXMACS also supports some of the most important styles used by publishing companies. For instance, Document ▸ Style ▸ Article ▸ Springer ▸ llncs can be used for articles in Springer's series of “Lecture Notes on Computer Science”.

Style files rely on *markup* to map specific intentions of the author to appropriate layouts. For example, it is customary to use either an italic or a bold font for important text. Such choices are made once and for all in the document style and then consistently applied throughout the entire document. We will return to this issue in the next section about document structure. For now, we observe once again that the consistent use of a minimal number of fonts and layouts is helpful in several ways: it allows the reader to stay focussed, without getting distracted by incongruous design decisions; it eases navigation within the document; and ideas can be transmitted through the mere use of appropriate typography.

The same lines of thought also apply to the main text. Many authors put a lot of effort into the science, some good will into the English spelling, and little energy into punctuation and other typographic issues. Fortunately, professional typesetting systems such as (LA)TEX and TEXMACS have taken over a large part of this task. However, despite sophisticated algorithms for spacing, line-breaking, page-breaking, positioning individual characters, etc., they occasionally require some human assistance.

Basic punctuation rules specify that you enter a space after a period "." at the end of a sentence, before starting a new one. If you correctly apply this rule, then TEXMACS will understand your intent and use this during the typesetting process (for instance, the space between two sentences will be slightly larger than between two ordinary words). Although TEXMACS does not forbid you to omit spaces after periods, it will fail to do the typesetting correctly if you break basic punctuation rules.

Explicit human intervention is typically required when using periods "." for other purposes, such as decimal dots, or inside abbreviations. Note that the space after an abbreviation is somewhat smaller than after the final period of a sentence. By identifying abbreviations through the use of appropriate markup, you may help TEXMACS to get the typesetting right.

Punctuation, typography, and even spelling are often perceived as dull and unimportant details. Please keep in mind that some of your readers might even if *you* don't. Your sloppiness may easily irritate or confuse others and lead to ambiguities or even errors. The good news is that it requires only minimal awareness and discipline to do things right in the first place. Furthermore, TEXMACS will do a big part of the job for you, as long as you follow a few basic rules. In sections 3.1 and 3.7, we will come back to these issues in more detail.

1.3 Structuring your documents

We have already stressed the importance of a consistent layout and the use of appropriate fonts. TEXMACS has been designed so as to make it as easy as possible to create so-called *structured documents*. When properly used, this ensures in particular a uniform visual appearance.

The aim of structured documents is to focus on *meaning* or *intent* rather than *presentation*. For example, instead of manually typesetting all section titles in a bold 17 point sized font, we urge you to use a special *markup element* section (also called *tag*) that is dedicated to section titles. This has many advantages: TEXMACS will automatically number your sections, guarantee a uniform presentation of all section titles, build an automatic table of contents, etc. Furthermore, whenever you change your mind, and wish to opt for another rendering of section titles, then it will not be necessary to modify the layout of every individual title in your document: it suffices to tell TEXMACS once and for all how to display the section tag.

The rendering of a document usually reflects its structure, which makes it natural to edit structured documents in a wysiwyg manner. For instance, it is likely that a short line, typeset using a bold 17 point font, and starting with a number, is always a section title. However, some of the structure may not directly be apparent: from the mere rendering of

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n, \tag{1.1}$$

it is not clear whether $\binom{n}{k}$ was entered as a vector or as a binomial coefficient. Indeed, if we were not aware of the existence of the appropriate binom tag, then we might have entered the binomial coefficient as a vector instead, which has a similar visual appearance. The use of appropriate tags becomes essential whenever the precise semantics of the formula matters. This happens for instance when you want to check the correctness of the formula, by copying it into your favorite computer algebra system. Another example concerns French translations, in which case the traditional notation for binomial coefficients becomes C_n^k.

Even in cases when it seems overkill to use structural markup for producing a certain visual effect, doing so remains a good habit. On the one hand, the systematic use of appropriate markup helps you to formalize what you are doing. On the other hand, structured documents are more sustainable, in the sense that they can more easily be reused in situations that you did not anticipate, initially.

The precise way in which markup elements are rendered is specified in so-called *style* files and packages. If you are new to TeXmacs, then you do not have to worry about how the actual translations take place: just select a style file that suits your needs and aesthetic preferences. The standard style files and packages offer a large number of markup elements that can be used in common situations.

More advanced users may actually want to define or customize some of the document structure by themselves. This typically happens when you introduce your own notations or if you frequently need a specific non-standard layout pattern. The simplest way to add a new markup element to TeXmacs is through the definition of a *macro*, using Tools ▸ Macros ▸ New macro. By default, such macro definitions are put in the *preamble* of your document. If you want to reuse the same macros in several of your documents, then you may group them together in your own style files or packages. See chapter 12 for more details.

The design of more and more complex macros requires some skill, but one nice aspect of the integrated nature of TeXmacs is that it is easy to study the way the built-in style files and packages work. For example, if your cursor is inside the binomial coefficient of formula (1.1), then the precise markup element binom that was used to produce this coefficient is displayed on the status and focus bars of your window (see Figure 2.1 on page 27), as well as in the Focus menu.

Using Focus ▸ Preferences ▸ Edit macro you may then edit the macro definition that specifies the rendering of the binom tag. You may also inspect the style package in which the binom tag is defined, using Focus ▸ Preferences ▸ Edit source. Through trial and error, this allows you to progressively learn how to write more and more sophisticated style packages by yourself.

1.4 The joy of wysiwyg-ness

In the same way as professional typesetters strive to produce documents that allow readers to fully concentrate on what they are reading, we feel that good text editors should allow authors to remain entirely focussed on the creative process of writing. Whereas readers may get distracted by typos, clumsy formatting, or loud fonts, the writing process gets interrupted whenever you are obliged to address technical details before you can enter your thoughts into the computer.

One prerequisite for maximal user comfort is to use a *wysiwyg* (what you see is what you get) editor. It is indeed most natural to edit documents in the actual form that you want to look in print; this is probably also closest to your mental representation of the document. However, mere wysiwyg-ness is not enough, since the precise way in which documents are being edited should also be both natural and efficient.

For instance, several basic mathematical formula editors propose ”palettes” with the available mathematical symbols, which tend to clutter the screen. If even simple symbols need to be entered *via* these palettes, then this also gives rise to a lot of back and forth movements between the keyboard and the mouse. Similarly, most front-ends to (LA)TEX force you to remember a long list of symbol names, or to interrupt the writing process just to look up the appropriate name of a symbol. In TEXMACS, a more efficient *and* user-friendly system was developed from scratch. Most symbols can be entered in an intuitive fashion, following a small number of basic rules (see section 4.2). For example, you may enter $\rightarrow$, $\mapsto$, α, $\nleqslant$, and $\subseteq$ via the keyboard shortcuts - >, | - >, a ⇥, < = /, and < ⇥ ⇥ =, respectively.

Historically speaking, early wysiwyg editors such as MICROSOFT WORD often lacked adequate tools for the efficient composition of scientific documents, especially those that contained many mathematical formulas. Moreover, their focus on visual rather than logical design often resulted in documents of poor quality. For this reason, such editors became a frequent target of mockery, “wysiwyg” being denigrated as “what you see is all you've got” [38, section 1.5]. As a result, the (LA)TEX movement took the rather dramatic step of abandoning the wysiwyg idea altogether. In (LA)TEX, documents are encoded in a textual pseudo-language, and a special compiler is required in order to obtain the desired printable versions. However, not everyone is comfortable using such a pseudo-language, which forces you to imagine the text you are typing, instead of actually seeing it.

Nevertheless, some of the historical criticism on wysiwyg editors is indeed justified and points to real problems. For example, the visual presentation of a document does not necessarily reflect its full structure, as we saw in the binomial coefficients example (1.1) of the previous section. But this drawback mostly vanishes for well written documents, under the reasonable assumption that the author (yourself) always used appropriate markup at the first place: if you *see* a binomial coefficient, then it *is* a binomial coefficient. In TeXmacs, the problem is further reduced by the fact that it is not only the final presentation on paper that matters, but rather the "full interaction" between the text and the author inside the editor. For instance, we already noticed that the structure becomes apparent when positioning the cursor inside. Similarly, one may select an alternative rendering style, and even a "LaTeX-source-like" rendering style in which the full structure is visually exposed.

Another issue concerns the ability to position the cursor at all relevant places inside the text. For example, consider the text "**bold***italic*". In most wysiwyg editors, there is a single cursor position available between the bold '**d**' and the first italic '*i*', which makes it difficult to guess whether an extra keystroke typed at this cursor position will produce bold, italic or plain text. By contrast, TeXmacs offers three cursor positions [24], one after the '**d**' inside the bold environment, one between the two environments, and one before the '*i*' inside the italic environment. The user knows exactly which of these positions the cursor is in, thanks to visual clues in the interface: in particular, the first englobing environment is displayed at the bottom-right of the window.

Cursor movement raises some other interesting problems in structured documents with formulas that heavily rely on two dimensional layout. For instance, some early structured wysiwyg editors would move the cursor according to the abstract structure. This typically implied that pressing the → button at the end of a numerator of fraction would move the cursor to the start of the denominator. It turns out that this way of doing it can be very confusing. In TeXmacs, the cursor movement is mostly graphical: if you press →, then you will go to the right, if possible [24, 1]. There are some exceptions, such as a cursor at the end of the line, or the cursor movement in computer algebra sessions, in which case it is natural to jump from one input to the other when moving up and down.

Other historical arguments against wysiwyg editors are more dubious. For example, some people believe that a "format" such as (LA)TeX is "more scriptable" in the sense that (LA)TeX documents can more easily be generated, combined and manipulated by external tools than TeXmacs documents. On a superficial level and for some very simple tasks, this may be so, because operating systems include many tools for the manipulation of text files. However, since TeXmacs documents are truly structured, it is unnatural to manipulate them as strings: one should rather regard them as expression trees, to be manipulated in a scripting language that can handle such objects. A large part of the TeXmacs interface is written in the SCHEME language, whose syntax and other features are particularly well suited for this task. Moreover, for those who really want to manipulate TeXmacs documents as text files, we offer several plain text formats, such as XML, SCHEME expressions, etc. The syntax of these formats is systematic and well-defined, contrary to (LA)TeX.

In summary, the most annoying drawbacks of early wysiwyg text editors have been addressed in TeXmacs through the implementation of various, occasionally novel, ideas. The further advantages of wysiwyg editors become most apparent for two dimensional layouts. For example, consider a table or a matrix. In TeXmacs, it is easy to insert a new column using a keyboard shortcut (⌥← or ⌥→), the menus, or the toolbars. In (LA)TeX or HTML, this requires you to modify every individual row of the table. Technical pictures are an even more convincing example. The original LaTeX `picture` environment invites you to enter the coordinates of each individual component. No wonder that (LA)TeX eliminated pictures from mathematical textbooks more surely than the BOURBAKI movement did!

1.5 TeXmacs as a structured editor

We have explained why structured documents help you to ensure a uniform presentation. We also discussed how TeXmacs allows you to efficiently write such documents, using a user-friendly, wysiwyg interface. You may even create new kinds of markup yourself, using the integrated macro language. Another important advantage of structure is that it leads to new opportunities for enhancing the editing experience.

One of the main ways to exploit structure inside the TeXmacs editor is through the Focus menu. The *current focus* refers to the innermost tag at the current cursor position. For example, assume that we positioned the cursor after the two in the following equation:

$$\frac{x^{\boxed{2}}}{y} = 1 + \frac{x}{y^3}. \tag{1.2}$$

The current focus is highlighted using a cyan box, whereas the grey boxes correspond to the other tags that contain the current cursor at higher levels.

The entries in the Focus menu allow you to directly perform editing operations on the current focus. For instance, in the example (1.2), the superscript can be turned into a subscript using Focus ▸ Superscript ▸ Subscript or the keyboard shortcut ^⇥. You may also jump to the next and previous scripts inside the document using Focus ▸ Next similar and Focus ▸ Previous similar (or ^⇟ and ^⇞), or obtain contextual help on the “superscript” tag rsup using Focus ▸ Describe. See sections 2.7 and 10.5–10.7 for more details.

The document structure is also taken into account for several common editing operations. For example, when searching “a” inside a mathematical formula, TeXmacs will only search for the symbol “a” inside other mathematical formulas. If you were editing a (LA)TeX source file, then such a search would typically result in many parasite non-mathematical “a” hits inside the main text. Using the TEXMACS search facility, it is even possible to find text in a structured way (see section 10.2). For instance, when searching for $\frac{x}{-}$, you will find any fraction in which x occurs in the numerator, such as

$$\frac{x}{y}, \quad \frac{1+x}{1+y}, \quad \frac{1+\sqrt{x}}{1-\sqrt{x}}, \ldots.$$

Another example of a structured editing facility concerns the computation, presentation and editing of the differences between two versions of a file (see section 10.9). As a general rule, the number of TEXMACS features that exploit document structure steadily grows with the years.

1.6 Towards a scientific office suite

Besides a scientific text editor, there are several other functionalities that can be expected from an office suite dedicated to science and education: the possibility to perform scientific computations from within the editor, an easy way to draw technical pictures, a spreadsheet facility, a laptop presentation tool, version control, collaborative authoring over the web, maintaining databases with bibliographic references, etc.

From early on, TEXMACS comes with interfaces for various computer algebra systems and other mathematical software. This allows you to perform complex mathematical computations directly from inside TEXMACS, while ensuring a professional rendering and a flawless integration of such computer algebra sessions into papers, books, and class material. Here is an example of a session inside the MAXIMA system:

(%i1) $\operatorname{diff}(x^{x^x}, x, 2)$

(%o1) $x^{x^x}(x^x \log(x)(\log(x)+1) + x^{x-1})^2 + x^{x^x}\left(x^x \log(x)(\log(x)+1)^2 + x^{x-1}\left(\log(x) + \frac{x-1}{x}\right) + x^{x-1}(\log(x)+1) + x^{x-1}\log(x)\right)$

(%i2) $\int \frac{x^5+2x-1}{x^2-3x+7} dx$

(%o2) $-\frac{57\log(x^2-3x+7)}{2} + \frac{37\arctan\left(\frac{2x-3}{\sqrt{19}}\right)}{\sqrt{19}} + \frac{x^4+4x^3+4x^2-60x}{4}$

Besides this traditional kind of interface, recent versions of TEXMACS also propose other mechanisms to interact with external software. For instance, any TEXMACS plug-in can be used as the computational engine of the built-in spreadsheet facility. Currently, it is also possible to use external software for silent computations in the background, such as evaluating or simplifying a selection while editing some formula. Finally, it is fairly easy to add your own plug-ins to external software. Chapter 11 is dedicated to the use of TEXMACS as an interface.

The technical picture editor is another extremely useful tool in the TeXmacs office suite. Although the tool is not as complete as dedicated editors for scalable vector graphics, it is well suited for simple images. One important advantage is that the tool is fully integrated, which makes it easy to put formulas or hyperlinks inside your pictures. Of course, it remains possible to embed pictures that were made using another tool into your documents. The technical picture editor is presented in detail in chapter 8.

A few decades ago, scientific presentations at conferences were done on a blackboard or using slides on an overhead projector. Nowadays, it is most common to directly project the screen of your laptop using a beamer. TeXmacs contains special markup for creating such laptop presentations, allowing you to progressively uncover content, play simple animations and sounds, compose complex overlays, etc. Various standard themes can be used for the general appearance and it is easy to add your own ones using the style package mechanism. It should be stressed that TeXmacs presentations are very dynamic: you may add new content or make corrections during your show. Computer algebra sessions can also be replayed in real time. See chapter 9 for more information about the presentation tool.

Recent developments aim at making TeXmacs more "web-aware". Indeed, when coauthoring papers with several people, it is fashionable to create a central repository on the web from where such papers are accessible to everyone and where to keep a record of all successive versions. SVN [6, 7] and GIT [62, 51] are well known versioning tools that allow you to accomplish this task. TeXmacs comes with an interface for SVN, and a special mode for going through the differences between various versions. Future plans exist for a centrally available TeXmacs server, with many other features beyond mere file sharing and version control.

1.7 Creating your own extensions

(LA)TeX [36, 38] and the GNU EMACS [53] text editor share the common strength that they make it easy for users to extend the system. (LA)TeX allows you to extend the system through the definition of new macros. This makes (LA)TeX quite different from other common markup languages such as HTML [41], where the allowed markup elements often form a fixed set, specified by a so-called DTD. The EMACS editor comes with a so-called *extension language* (called EMACS-LISP, a special dialect of LISP), which allows you to customize the editor and to program new features. One common application is to add support for a new programming language (e.g. syntax highlighting, automatic indentation, etc.).

TeXmacs offers several similar mechanisms for customizations and extensions. We already mentioned the possibility of defining new macros and bundle such macro definitions together into style files and packages (see chapters 12 and 13). TeXmacs uses SCHEME instead of EMACS-LISP as its extension language. All high level editing routines are written in this language and new routines can easily be implemented by the user. In particular, the TeXmacs menus, keyboard shortcuts, and most elements of the graphical interface are programmed in SCHEME. Besides, certain TeXmacs macros may occasionally call SCHEME routines for computing their rendering. The last chapter 14 of this book contains an introduction to SCHEME programming for TeXmacs.

TeXmacs offers one more extension mechanism, which is also used for the creation of interfaces with external software. A TeXmacs *plug-in* consists of a directory with a collection of various files: TeXmacs style files and packages as above, SCHEME files with customizations or extensions of the editor, TeXmacs files with useful documentation, and any other files of interest, such as communication scripts with external programs. TeXmacs plug-ins can easily be shared among users and it suffices to put them at a dedicated place in order to activate them. For more details, we refer to the integrated documentation in Help ▸ Interfacing.

1.8 TeXmacs and (LA)TeX

Before anything else, we stress that recent versions of TeXmacs are completely independent from (LA)TeX [36, 38, 16]. Contrary to what its name might suggest, TeXmacs is *not* a (LA)TeX front-end.

Initially, the development of TeXmacs got some of its inspiration from (LA)TeX and EMACS, whence its name. For instance, TeXmacs uses the same fonts as TeX, and borrows several of its typesetting algorithms (such as the way paragraphs are hyphenated and broken into separate lines). The graphical interface of older versions of TeXmacs looked similar to the EMACS interface, especially concerning keyboard shortcuts and the menus. Finally, as mentioned in the previous section, TeXmacs offers similar extension mechanisms as (LA)TeX and EMACS.

Over time, the differences between TeXmacs, (LA)TeX, and EMACS have become more important. Recent versions of TeXmacs strive to use the same graphical user interface conventions as other software on your system, although you may still opt for an EMACS look and feel in the preferences. TeXmacs has also continued to grow out into a complete scientific office suite, whose capacities go far beyond the production of good looking scientific documents (see section 1.6).

Obviously, the major advantages of TeXmacs with respect to LaTeX are its higher versatility and its user-friendly, wysiwyg graphical user interface. The integrated drawing tool and presentation mode are highly convenient as well. There are also a few less prominent improvements. For example, TeXmacs uses better algorithms for page breaking and the computation of space between successive lines. Mathematical formulas also carry more semantics than in LaTeX, which is important for interfacing purposes with external software. For example, (LA)TeX makes no essential distinction between the formulas $f(x+y)$ and $a(b+c)$. However, we probably meant "f applied to $x+y$" versus "a multiplied with $b+c$"; TeXmacs encourages you to make the intended meaning explicit.

When it comes to more specialized features, (LA)TEX occasionally conserves its advantages over TEXMACS. Due to its large user base, there are many packages for exotic typesetting needs: if you are writing an essay on Egyptian mathematics and wish to combine mathematical formulas with hieroglyphs, then you will still need (LA)TEX. If you have special ideas on how to present algorithms, then you may be lucky to find a LATEX package that implements precisely your point of view. A few useful LATEX features have also not yet been implemented in TEXMACS, such as page breaking of long tables and wrapping text around figures. Of course, such features are on the TEXMACS developer's wish lists, so it will only be a matter of time before they will also appear in TEXMACS.

The main advantage of LATEX with respect to TEXMACS is of a social nature: most publishers require scientists to send final versions of papers in LATEX and many professors force their students to learn LATEX, sometimes even through dedicated courses. This is a sad situation for many of our users. Fortunately, PDF is an even more standard format nowadays, so we recommend users to systematically push publishers towards accepting either TEXMACS or PDF documents. For those who do not need to collaborate with colleagues who use LATEX or send their work to publishers, the above “advantage” of LATEX vanishes. This holds in particular for high school education or for less formal documents in academia.

Besides its lack of wysiwyg-ness, (LA)TEX also has another major drawback: there does not exist such a thing as a (LA)TEX document format. In fact, (LA)TEX is really a programming language, but, unlike other programming languages, (LA)TEX does not obey any well-defined formal syntax. This means that (LA)TEX documents can essentially only be processed in a reliable way by (LA)TEX itself. For instance, it is easy to write a (LA)TEX package in which $*$ means “plus” on even pages and “start a new theorem” on odd pages.

The lack of a well-defined (LA)TEX format has caused endless headaches and frustration for the TEXMACS developers, since it makes it impossible to write 100% reliable converters between LATEX and other formats, despite a huge amount of work in this direction. Unfortunately, few LATEX users are aware of this fact and its corollary that their “source code” might be of little use within ten or twenty years from now. It is also plausible that this “locking up in the LATEX non-format” effect has slowed down the development of other mathematical software besides TEXMACS.

Although TEXMACS comes with high quality converters from and to LATEX, the above discussion shows that one cannot consider these mechanisms as 100% reliable black boxes. For those who frequently need to convert documents between TEXMACS and LATEX, there exist various conversion options so as to make the conversion process as smooth as it can get.

Chapter 2
The user interface

The graphical user interface of $\mathrm{T_EX_{MACS}}$ has been designed to be as intuitive as possible. In particular, recent versions of $\mathrm{T_EX_{MACS}}$ strive to respect standard conventions that are also used by other software on your computer: basic keyboard shortcuts will be the same and parts of the menus are organized similarly. It is therefore likely that you will learn much of $\mathrm{T_EX_{MACS}}$ just by using it. Nevertheless, for getting the most out of the software, it may be useful to be aware of some general design principles and conventions that are used inside $\mathrm{T_EX_{MACS}}$ and this book.

Of course, explanations about user interfaces are easier to remember when you start putting them into practice. You may therefore wish to skip some of the details when reading this chapter for the first time. Once you understand the basics, you may use this chapter as a reference and return to it for more specific questions about the $\mathrm{T_EX_{MACS}}$ user interface.

2.1 Basic principles

First of all, different users have different backgrounds and distinct preferences when it comes to ergonomics. As long as individual habits do not conflict with each other, $\mathrm{T_EX_{MACS}}$ strives to support as many of them as possible. Take the example of a simple editing action like the insertion of a new section title. If you are new to $\mathrm{T_EX_{MACS}}$, then the Insert ▸ Section menu or the ≣ icon is the natural place to look at. However, if you frequently need to insert new sections, then it is more convenient to use a keyboard shortcut. The appropriate one is `⌥1`, as indicated in the Insert ▸ Section menu. Finally, if you already know LaTeX, then you may prefer to use the `\section` command. For this, simply type `\` `s` `e` `c` `t` `i` `o` `n`, followed by enter `↩`. The `\`-key is heavily overloaded in $\mathrm{T_EX_{MACS}}$ and can also be used in order to enter other markup elements that do not necessarily exist in LaTeX.

A second basic principle is that $\mathrm{T_EX_{MACS}}$ tries to be as *contextual* as possible. If we permanently advertised all features of $\mathrm{T_EX_{MACS}}$ on your screen, then blinking buttons would soon drive you crazy. Many features therefore only appear when they are most relevant. One problem with this is that you may miss certain functionalities if you are not looking at the right place. In order to reduce this risk, the interface contextualization is governed by a few simple rules:

1. On the most global level, you may enable or disable certain features in the user preferences or using the Tools menu.

2. On an intermediate level, many of the relevant editing actions depend on the *current editing mode*, which reflects the kind of object you are manipulating: text, mathematics, graphics, programs, etc. There are only a very limited number of editing modes, but they greatly affect the general behavior of the editor.

3. On the finest level, many structured editing actions depend on the *current focus*. The current focus corresponds to the innermost tag at the current cursor position. Since TEXMACS defines hundreds of different markup elements, the Focus menu and other focus-dependent editing actions are highly contextual.

Imagine for instance that we wish to enable a European numbering style of theorems inside TEXMACS (meaning that theorems, propositions, etc. are numbered individually instead of sharing a common counter). In this case, a search for "European numbering" in the documentation (using F1 or Help ▸ Search ▸ Documentation) will lead you to the relevant place. However, it is likely that the very terminology "European numbering style" is new to you. In order to find the appropriate information in that case, the idea is to first insert a theorem, using Insert ▸ Enunciation ▸ Theorem. The contextual Focus ▸ Preferences menu then contains an item European numbering style that you may try out and hope that it does what you want.

2.2 The main window

After starting a new document, the main TEXMACS window will usually look similar to the screenshot in Figure 2.1 below. The screenshot was taken on a computer that runs MACOS, so it might look slightly different on other systems. Furthermore, the TEXMACS menus are displayed at the top of the screen under MACOS, rather than in the window itself, as under GNU/LINUX or WINDOWS.

Let us briefly describe the components of the main TEXMACS window:

- The *title bar* displays the name of your document and also indicates whether the changes you made in your document have been saved on disk.

- The *main icon toolbar* contains context-independent icons that are generally useful (opening, saving, and printing files, opening the user preferences, closing TEXMACS, copying and pasting, search and replace, undo and redo, navigation buttons). Most of the icons correspond to entries in the File, Edit, and Go menus.

- The *mode-dependent icon toolbar* varies with the current editing mode and contains different icons when you are editing mathematical formulas or graphics instead of plain text. Most of these icons are used for inserting new markup elements into your document and correspond to entries in the Insert menu.

- The *focus toolbar* contains icons that depend on the current focus. This toolbar frequently changes when moving your cursor around in a structured document. Most of the icons correspond to entries in the Focus menu.

- The *status bar* contains useful information. The *left footer* generally shows the current mode and font properties, whereas the *right footer* indicates the structure at the current cursor position. The status bar is also used for certain warnings, error messages and interactive input from the user.

- The *search toolbar* is not displayed by default, but gets activated when starting a new search. The bar can be *expanded* into a separate window by pushing the ▭ button and closed by pushing ⊗. The *replace* and *spell toolbars* work in a similar way and it is likely that future versions of TeXmacs will implement more and more toolbars of this kind.

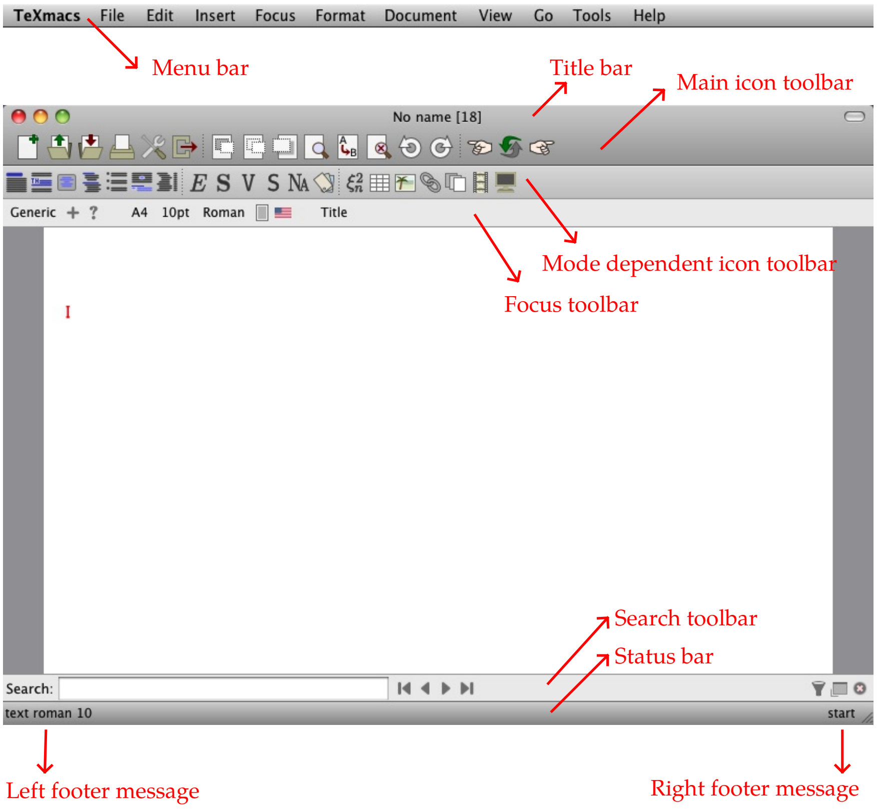


Figure 2.1. The main TeXmacs window. On most computers, the menu bar is part of the TeXmacs windows. Some operating systems such as MACOS rather display them at the top of the screen.

Key	Meaning	EMACS	GNU/LINUX	WINDOWS	MACOS
⇧	Shift	S-	Shift	Shift	shift
^	Control	C-	Ctrl	Ctrl	control
⌥	Alternate	A-	Alt	Alt	option
⌘	Meta	M-	Meta	Windows	command

Table 2.1. Notations for modifier keys in this book.

Not all icon toolbars are always visible. Their visibility can be controlled from within the View menu. It is also useful to know that most of the icons on the toolbars are self-explanatory: when keeping your mouse over the icon during a second or so, a help balloon with a short description of its purpose will appear. The help balloon also indicates an equivalent keyboard shortcut if there exists one.

2.3 Mastering the keyboard

Many editing operations in TEXMACS can most efficiently be executed using *keyboard shortcuts*. There is an abundance of such shortcuts in TEXMACS and new ones can be defined by the user. Before we present general principles that will help you to learn and remember them, let us start with explaining how they will be represented in this book.

2.3.1 Notation for keyboard shortcuts

Keyboard shortcuts are combinations of keys that are pressed either concurrently or in a sequence. In the case when several keys are pressed at the same time, all but one of the keys being pressed are so-called *modifier keys*. Such modifier keys are somewhat larger than the other keys on your keyboard, so your fingers can find them easily. TEXMACS uses four modifier keys: Shift ⇧, Control ^, Alternate ⌥, and Meta ⌘. Most keyboards contain Shift, Control (or Ctrl), and Alternate (or Alt) keys. The "Meta" key usually carries another name, which depends on your keyboard model and operating system. The naming conventions for different keyboard models and operating systems are summarized in Table 2.1.

We represent keyboard shortcuts by small boxes that graphically recall the idea of a key on your keyboard. When several keys are displayed in the same box, they are assumed to be pressed at the same time. Otherwise, we assume that you press them in succession. For instance, ⌘⇧a stands for "simultaneously press the three keys ⌘, ⇧, and a". The shortcut - > means "first press -,

Key	Meaning	Key	Meaning
⇧	Shift modifier	←	Cursor left
⇪	Caps lock	→	Cursor right
^	Control modifier	↑	Cursor up
⌥	Alternate modifier	↓	Cursor down
⌘	Meta modifier	↖	Home
↩	Return (or Enter)	↘	End
⌦	Forward delete	⇞	Page up
⌫	Backspace	⇟	Page down
⎋	Escape	␣	Space
⇥	Tab		

Table 2.2. Notations for special keys in this book.

then release -, next press >". Similarly, ^x ^f stands for "simultaneously press ^ and x, then release both keys (or at least x), next simultaneously press ^ and f.

It is convenient to extend the short graphical representations ⇧, ^, ⌥, and ⌘ for the modifier keys to other special keys on your keyboard. Table 2.2 summarizes the notations that are used in this book.

2.3.2 How to remember keyboard shortcuts

Unfortunately, there are far more useful editing operations than keys on your keyboard. This raises the question of how to associate keyboard shortcuts to editing actions in a way that is easy to remember. TeXmacs achieves this through a mixture of ideas.

First of all, recent of versions of TeXmacs try to respect as much as possible standard conventions used by other software on your computer; see also section 2.3.6. The most common keyboard shortcuts that you already learned for other programs should therefore also work inside TeXmacs.

Secondly, many editing operations are similar or at least share one common characteristic. For instance, we may single out the class of "structured insertions/removals". TeXmacs attempts to group shortcuts for operations in such a class together via a distinctive prefix, modifier key, or modifier key combination. Structured insertions and removals are assigned the common modifier key ⌥, with the consequence that you may insert a new column in a table or a new argument in a list of bibliographic citations using ⌥← or ⌥→. You may also perform the corresponding removals of columns and citations using ⌥⌫ and ⌥⌦.

The next idea is based on the concept of "variants". In the case of a list of similar editing operations, you may use the Tab-key ⇥ to cycle among the different variants. This mechanism is highly useful for entering mathematical symbols: $\in$, $\subset$, and $\prec$ are examples of variants of $<$, which are entered using < ⇥, < ⇥ ⇥, and < ⇥ ⇥ ⇥. The variant mechanism is also used when several editing operations compete for the same key. For example, inside mathematics, we use the keyboard shortcuts ^ and _ for entering superscripts and subscripts, as does (LA)TeX. Inside text, these shortcuts rather insert the symbols "^" and "_", yet superscripts and subscripts can be obtained through the shortcuts ^ ⇥ and _ ⇥. The Tab-key ⇥ is highly versatile: it is also used for "Tab-completion" of words and the selection of the appropriate object in the graphical mode, whenever ambiguities arise through superposition. The Tab-key also produces interesting effects when combining it with modifier keys: ⇧⇥ allows you to circle back, whereas ^⇥ is used for "structural variants" such as turning a theorem into a proposition.

There are a few other noteworthy mechanisms that are mode-specific. In math mode, one such mechanism is "graphical concatenation", which allows you to type $\rightarrow$ using - >, $\Longleftrightarrow$ using < = = >, and $\leqslant$ using < =. We refer to section 4.2 for further examples and tables with mathematical symbols. In text mode, accented characters are formed using the ⌘ modifier key (e.g. ⌘' e yields "é") and special symbols are obtained using the prefix ⇧F5 (e.g. ⇧F5 s yields ß): see section 3.1.

One should also be aware of the fact that the TeXmacs keyboard behavior is highly contextual: it depends on the user preferences, the current mode, as well as the current focus. For instance, inside "math mode", you have special keyboard shortcuts that are handy for typing mathematical formulas, but which are useless in text mode. This allows for a substantial amount of overloading, in the sense that the same keyboard shortcuts serve different purposes in different contexts. For example, after activating a computer algebra system as a scripting language (using Document ▸ Scripts), you may use ^↩ in order to evaluate a mathematical formula. When going through the differences between two TeXmacs documents using the versioning tool, the same shortcut ^↩ is rather used for retaining a selected version.

We finally recall that menu entries indicate equivalent keyboard shortcuts whenever they exist. The keyboard equivalent for an icon on the toolbars appears in a popup menu, when hovering the mouse pointer over the icon for a while. In math mode, the same holds for the submenus with "palettes" of mathematical symbols.

2.3.3 Hybrid commands and LaTeX emulation

TeXmacs uses \ as an important multi-purpose key. By hitting the \ key, you enter the hybrid LaTeX/TeXmacs command mode. As soon as you finished typing a LATEX or TEXMACS command, the left footer displays something like

```
<return>: action to be undertaken
```

At this stage, hitting the return-key ↩ will execute your command. For instance, in math-mode, typing \ f r a c ↩ creates a fraction.

Technically speaking, TEXMACS first checks whether you typed a LATEX command. If your command is not recognized as a LATEX one, then TEXMACS looks whether it is an existing TEXMACS command, macro argument or environment variable (provided by the style file; see chapters 12 and 13). If so, the corresponding macro expansion, macro argument or environment variable is inserted, with the right number of arguments. The \ key is always equivalent to one of the more specific shortcuts ⌘i l (LATEX command), ⌘i c (TEXMACS command), ⌘i a (macro argument), or ⌘i v (environment variable).

To insert a literal \ (backslash), you can use ⇧F5 \ or \ ⇥.

2.3.4 Common modifier combinations and prefix rules

We have seen that shared modifier combinations or prefixes make it easier to remember keyboard shortcuts for editing operations of a similar nature. The precise behavior heavily depends on the selected "look and feel" in Edit ▸ Preferences ▸ General. I personally own a MAC laptop, on which I use the EMACS look and feel. Under these circumstances, the most common modifier combinations and prefixes are as follows:

^. Standard shortcuts, that are similar to shortcuts used by other applications (for the selected look and feel). For example, ^y can be used for pasting text (as in EMACS).

⌥. TEXMACS shortcuts that may rely on the current editing mode. For example, ⌥s produces **strong** text in text mode and a square root $\sqrt{}$ in math mode.

⌘. Compound TEXMACS shortcuts. Usually, these shortcuts first indicate the kind of markup to which the command will apply and then specify the specific command. For instance, the ⌘e prefix is used for inserting executable markup, which is useful for writing style files (see chapter 13). One example is the shortcut ⌘e + for inserting an addition.

⌘. This modifier key is used in combination with arrow keys and certain other special keys for positioning and resizing objects (see section 10.8).

⌘⌥. This modifier combination is used in combination with arrow keys and some other special keys for structured cursor movements (see section 10.6).

Purpose	EMACS	GNOME	KDE	WINDOWS	MACOS
Standard	⌃	⌃	⌃	⌃	⌘
Mode-specific	⌥	⌥	⌥	⌥	⌃
TeXmacs-specific	⌘	⌘	⌘	⌘	⌘⌃
Position & resize	⌘	⌘⌃	⌘⌃	⌘⌃	⌘⌥
Structured navigation	⌘⌥	⌘⌥	⌘⌥	⌘⌥	⌘⌃
Extra	⌘⌃	⌥⌃	⌥⌃	⌥⌃	⌥⌃
Special symbols	⇧F5	⇧F5	⇧F5	⇧F5	⇧F5

Table 2.3. Common prefix rule correspondences for different look and feels. Recall that the shortcuts in this book correspond to the EMACS look and feel.

⌘⌃. This modifier combination is occasionally used in combination with letters and punctuation symbols for creating some additional easy to remind shortcuts.

⇧F5. This prefix can be used in combination with normal letters for the insertion of special symbols. For example, ⇧F5 s yields ß and ⇧F5 a yields Ŀ. The ⇧F5 prefix is also used for the insertion of "literal characters". For instance, ⇧F5 \ will always produce the \ character, whereas the \ key is used for entering hybrid commands.

If you are using a different computer or look and feel, then some of these prefix rules may change: see Table 2.3.

2.3.5 Substitution rules

On certain keyboards, important modifier keys such as the "Meta" key (or a substitute) are sometimes missing, or reserved for other purposes. For instance, ⌥-based shortcuts are used for entering accented characters and common special symbols under MACOS. Certain important TeXmacs keyboard shortcuts can also be superseded by shortcuts from the operating system. For example, the MACOS application SPACES uses the shortcuts ⌃←, ⌃→, ⌃↑, ⌃↓, ⌃1, ⌃2, ⌃3, and ⌃4 to switch between multiple screens.

One solution to the second problem is to change the problematic global shortcuts that got overridden in the responsible applications. For example, SPACES can be configured to use ⌘⌥⌃ as a prefix instead of ⌃ (click on the popup menu behind "To switch between spaces" and simultaneously press ⌘, ⌥, and ⌃). Notice that fn is another key that is not used by TeXmacs.

If you cannot or do not want to change the system-wide shortcuts, or if one or more modifier keys are missing on your keyboard, then you may use the

Shortcut	**EMACS**	GNOME	KDE	WINDOWS	MACOS
⎋	⌘	⌘	⌘	⌘	⌥
⎋ ⎋	⌥	⌥	⌥	⌥	^
⎋ ⎋ ⎋	^	^	^	^	⌘
⇧⎋	⌘⌥	⌘^	⌘^	⌘^	⌘⌥
⇧⎋ ⇧⎋	⌘^	⌥^	⌥^	⌥^	⌥^
⇧⎋ ⇧⎋ ⇧⎋	⌥^	⌘⌥	⌘⌥	⌘⌥	⌘^

Table 2.4. Keyboard shortcuts for modifier keys or modifier key combinations. Recall that the shortcuts in this book correspond to the EMACS look and feel.

Escape-key ⎋ in order to produce equivalents for the modifier keys ⌘, ⌥, and ^. For instance, under MACOS, ^ is equivalent to ⎋ ⎋. Hence, the TeXmacs shortcut ^→ can also be obtained by typing ⎋ ⎋ →, which may coexist with the SPACES shortcut ^→. Table 2.4 shows the modifier key combinations that can be obtained using ⎋.

2.3.6 Standards conformance

TeXmacs attempts to be as standards-conforming as possible for each of the supported look and feels in Edit ▸ Preferences ▸ General. However, there are a few general situations in which TeXmacs reserves keyboard shortcuts for the sake of user-friendliness:

- The function keys F5 – F12 are reserved for special actions.
- Most operating systems admit a "principal modifier key" for forming keyboard shortcuts (e.g. ^ for the EMACS look and feel) as well as a "secondary modifier key" for a few other shortcuts (e.g. the `Windows` key under WINDOWS and ⌥ under MACOS). The remaining free modifier key (e.g. ⌥ for the EMACS look and feel) is reserved for TeXmacs.
- TeXmacs contains many keyboard macros involving one or more modifier keys and the special keys ←, →, ↑, ↓, ↖, ↘, ⇞, ⇟, ⌫, ⌦, ␣, ⇥, and ↩. The behavior of shortcuts of this kind is occasionally non-standard.

The TeXmacs-specific shortcuts are rarely in conflict with standard conventions. Nevertheless, in Table 2.5, we have listed a few more or less standard shortcuts, that might work in other applications, but that will usually not work inside TeXmacs.

Look & feel	Shortcut	Alternative	Meaning
Emacs	F10		Show menu bar in window
Emacs	⌘!		Shell command
Emacs	⌘' / ⌘` / ⌘∧		Needed for TeXmacs accents
Emacs	⌘/ / ⌘\ / ⌘: / ⌘;		
Emacs	⌘← / ⌘→	^← / ^→	Move word back/forward
Emacs	⌘a / ⌘e	^↑ / ^↓	Move paragraph back/forward
Emacs	⌘b / ⌘f	^← / ^→	Move word back/forward
Emacs	⌘l / ⌘t		Locase/transpose words (not impl.)
Windows	F5		Refresh/Switch to next pane
Windows	F6 / ^F6 / ^⇧F6		Switch to next/previous pane/tab
Windows	^␣		Remove formatting
Windows	^⇥		Switch to next child window
Windows	^⌫ / ^⌦		Delete word
MacOS	^F5 / ^F6 / ^⇧F6		Move focus to toolbar/panels
MacOS	^F7		Override keyboard access mode
MacOS	F9 / F10		Tile or untile windows
MacOS	F11 / F12		Hide or show windows/dashboard
MacOS	⇥ / ⇧⇥		Navigate through controls
MacOS	^⇥, ^⇧⇥		Move focus within control groups
MacOS	^␣ / ⌘^␣		Toggle between input sources
MacOS	^← / ^→	⌘⌥← / ⌘⌥→	Move one cell left/right in table
MacOS	^↑ / ^↓	⌘⌥↑ / ⌘⌥↓	Move one cell up/down in table
MacOS	↖ / ↘	⌘↑ / ⌘↓	Move to start/end of document
MacOS	⌥⇞, ^↑, ^⇞	⇞	Page up
MacOS	⌥⇟, ^↓, ^⇟	⇟	Page down
MacOS	^a / ^e	⌥↑ / ⌥↓	Move to start/end of block

Table 2.5. Shortcuts that might work in other applications, but usually not in TeXmacs.

2.4 Organization of the menus

Recall that we use a sans serif font for menu entries in this book. For instance, the menu entry for creating a new file is denoted by File ▸ New. The top-level menus of TeXmacs were designed according to the following principles:

- The File menu contains the most common operations on files: opening, saving, closing, and printing of documents, as well as conversions to other formats.
- The Edit menu is also pretty standard and contains the most basic editing operations: copying and pasting of text, undoing and redoing changes, search and replace, as well as spell checking. TeXmacs also allows you to copy and paste fragments of documents that use other formats. For example, when copying and pasting a selection from your favorite web browser, most of the formatting properties will be preserved.

- The Insert menu is mode-dependent and used for the insertion of structural markup. In text mode, it allows you for example to insert section titles, theorems, numbered lists, emphasized text, etc. In math mode, it allows you to insert fractions, wide accents, matrices, etc.

- The Focus menu corresponds to editing operations that act on the current focus; see section 2.7. Assuming that your cursor is inside a section title, it allows you to transform the section into a subsection, to toggle the numbering, to jump to other sections, etc.

- The Format menu is dedicated to visual fine-tuning of the layout and presentation of pieces of your document. You may change the font, the color, the alignment and spacing of paragraphs, insert extra whitespace, control line and page breaking, adjust the position and size of pieces of text, etc.

- The Document menu allows you to specify the global properties of your document, such as its style and language, the page size and color, the main font, metadata, etc.

- The View menu is used for controlling the presentation of the TeXmacs windows. It specifies which toolbars are visible and the zoom factor that is applied to your document. It also allows you to enter and leave presentation mode.

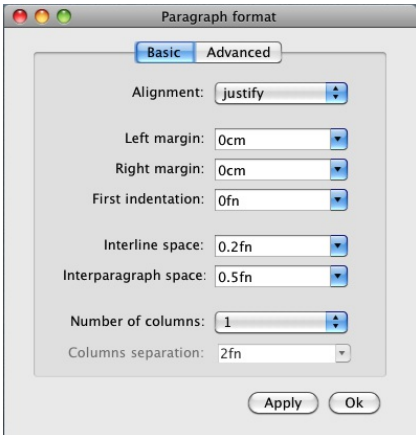
- The Go menu allows you to change the document that is visible in the main window. Its Back and Forward entries can be used for browsing a collection of TeXmacs files, such as the built-in documentation.

- The Tools menu contains miscellaneous utilities. It also allows you to enable additional tools for debugging, code development, versioning, etc. When enabling the versioning tools, for example, a new top-level Version menu will appear with additional document versioning facilities.

- The Help menu gives access to the built-in documentation. TeXmacs includes a manual, a reference guide, and more technical information about its implementation. It is also possible to search for specific keywords inside the documentation.

2.5 Dialogue windows

In old versions of TeXmacs, most of the complex editing operations were accomplished through the menus. More recently, a big part of the user interface has been redesigned so as to make more extensive use of dialogue windows. For instance, fonts can be picked through a special font selector and the page layout can be modified using a dedicated widget.



Alignment: justify

Left margin: 0cm

Right margin: 0cm

First indentation: 0fn

Interline space: 0.2fn

Interparagraph space: 0.5fn

Number of columns: 1

Columns separation: 2fn

Figure 2.2. The dialogue window for changing the layout of paragraphs.

In order to explain how dialogue windows work, it is convenient to extend the notation that we introduced for menus. Let us see how this works on an example. In Figure 2.2 below, we have displayed the dialogue window for modifying the layout of paragraphs. This window is opened using Format ▸ Paragraph. Then Basic ▸ Alignment ▸ justify corresponds to the selection of justify for the paragraph alignment in the list of choices next to Alignment. The notation Format ▸ Paragraph ▸ Basic ▸ Alignment ▸ justify stands for the combined action of opening the dialogue window and selecting the justified paragraph alignment.

Some users do not like their screen to be cluttered by dialogue windows and prefer to use the menus as much as possible for complex editing operations. TeXmacs provides a special user preference for this: Edit ▸ Preferences ▸ General ▸ Complex actions ▸ Through the menus.

2.6 User preferences

Many functionalities of TeXmacs can be configured according to the preferences of the user. Some of the most important user preferences can be changed in Edit ▸ Preferences. However, many settings are relevant for specific tools or markup elements only. Changing such settings is done in the preference menu of the corresponding tool or in an appropriate context-dependent preference menu. Some of your likings are even determined automatically by TeXmacs. For example, the size and position of the TeXmacs window are remembered when closing and restarting the application.

In Edit ▸ Preferences ▸ General, you will find a few important global user preferences that control the general behavior of the user interface:

Look and feel. Each particular operating system comes with its own user interface conventions for common editing operations like "cut and paste". By default, TeXmacs tries to conform to the conventions of your operation system, but you are free to choose another "look and feel". You may also choose the Emacs look and feel in order to enable many of the GNU EMACS keyboard bindings.

User interface language. Whenever appropriate translations are available, TeXmacs will use the specified language for the user interface (e.g. the menus, dialogue windows, etc.).

Complex editing actions. By default, TeXmacs opens dedicated dialogue windows for accomplishing complex editing operations. Some users prefer not to clutter their screen with other windows and rather perform such operations through the menus.

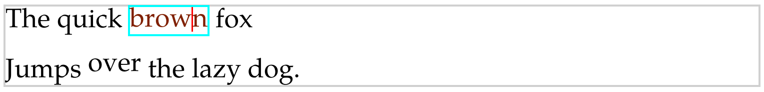
Interactive questions. Some editing operations require extra input from the user. Such additional feedback can either be entered in popup windows or via the footer.

Details in menus. TeXmacs offers many possibilities. For some applications (e.g. high school education), one may wish to simplify the user interface by only exposing a subset of it.

Buffer management. Many applications create a separate window for every new document that you open. Some users prefer the GNU EMACS convention of using a single window in which you may switch between multiple documents, also called *buffers*.

2.7 The current focus

TeXmacs offers a wide range of features. The main mechanism through which only the most relevant functionalities in a given context are exposed to the user is based on the concept of *current focus*. Recall that TeXmacs documents are *structured*. At a given cursor position, we find ourselves inside a list of structures (also called *tags* or *markup elements*), ordered from innermost to outermost. For instance, assume that our cursor is behind the "w" of "brown" inside the following quotation:


The quick brown fox

Jumps over the lazy dog.

Then we are inside two structures: a change of color into brown and a quotation. TeXmacs displays this information in the right footer. It also indicates the innermost structure through a cyan box and all other surrounding structures through light grey boxes. The innermost structure at the current cursor position is called the *current focus*.

Remark. When selecting text, the current focus rather becomes the innermost structure that contains the selection. Here it is useful to know that the ^␣ keyboard shortcut enlarges the current selection to its surrounding structure. Through the repeated use of this shortcut, any of the tags that contain the cursor can thus be selected efficiently and act as the *focus tag*.

The main editing operations that correspond to the current focus are accessible through the Focus menu and the *focus toolbar*. The items in this menu and on this toolbar are highly context-dependent by design. Whenever appropriate, they allow you to perform the following types of actions:

Navigation. The icons ⏶, ▲, ▼, and ⏷ (which correspond to the menu entries First similar, Previous similar, Next similar, and Last similar) can be used for jumping to similar tags in your document. Assuming that you are inside a theorem, clicking on ▼ will for example bring your cursor to the next theorem, proposition, lemma or corollary.

Insertion and removal. Certain tags (such as bibliographic citations, "switches" in presentations, tables, etc.) admit a variable number of arguments. Using the icons ⊕ and ⊕ (or the menu items Insert argument before and Insert argument after), you may add new arguments. Similarly, ⊗ and ⊗ (or Remove argument before and Remove argument after) can be used for the removal of arguments. In the case of tables, the above icons (or menu entries) can be used for the insertion and removal of columns. In addition, you may use ⊕, ⊕, ⊗, and ⊗ for the insertion and removal of rows.

Variants. Certain tags admit variants or alternative states. Theorems can be numbered or not for example and they behave much alike propositions, lemmas and corollaries. Other markup elements can be folded or unfolded (this is useful for presentations and exercises). Now the focus menu and toolbar always indicate the name of the tag that corresponds to the current focus. If this tag has interesting variants, then a submenu with the list of variants is proposed under its name. Similarly, the icons Ⅳ, ◀, ◀, and ◁ can be used to toggle the numbering or state of the tag, whenever applicable.

Exit. The icons ⇤ and ⇥ (or Exit left and Exit right) can be used in order to exit the focus tag at the left or the right.

Destroy. The focus tag can be destroyed using ✗, Delete or ^⌫. More precisely, the tag and its arguments will be replaced by the argument that contains the cursor.

Customize. The presentation of the focus tag can be customized via 🔧 or Preferences. Notice that customizations of this kind will act on the style properties and preamble of your document and not on the user preferences, strictly speaking. In other words, alternative settings will only impact on the document that you are editing and not on any other documents that you may open in the future. The following types of customizations are most common:

- Some tags come with one or more *style options* that correspond to customized renderings through the use of special style packages. For example, theorems may be told to use an alternative "European numbering style" using Focus ▸ Preferences ▸ European numbering style.

- Various *style parameters* (also called *environment variables*) may influence the rendering of a tag. For example, list environments come with two style parameters *item-hsep* and *item-vsep* that control the horizontal and vertical spacing of list items. The Preferences menu comes with an entry for specifying each style parameter of the focus tag.

- All non-built-in TeXmacs tags are defined using macros in the TeXmacs style packages or directly by the user. If applicable, then you may use Focus ▸ Preferences ▸ Edit source in order to jump to the relevant place in the style package that defines the focus tag. You may also edit the corresponding macro inside a special dialogue window using Focus ▸ Preferences ▸ Edit macro; see section 12.1 for more information.

Documentation. Contextual help about the focus tag can be obtained using ? or Describe.

Arguments. Certain tags come with one or more "hidden" arguments that are invisible in the document itself and therefore require special mechanisms for being edited. Typical examples of hidden arguments are the color "brown" in the example at the beginning of this section or the names of labels and references. In many cases, such hidden arguments are short names, colors or lengths, that can be edited via extra input fields on the focus toolbar. In general, tags with hidden arguments can be deactivated and reactivated using ☑ or Show hidden. Alternatively, you may place the cursor just behind the tag and press ⌫ in order to deactivate it. After editing the hidden arguments, you may reactivate the tag using ↩.

In the case when the current cursor is not inside any particular tag, the current focus is on the whole document. This happens for instance when the cursor is at the extreme top-left position in the document. Some of the most important entries of the Document menu are then exposed more directly on the focus toolbar:

Style. On the extreme left, you may specify the style of your document, and select any additional style packages to be used. We recall that style packages that are really style options for particular tags should rather be specified via the Focus ▸ Preferences menus for these tags.

Page style. The current page size is indicated on the focus bar and can be modified. Similarly, the orientation and number of columns are indicated through one of the icons ▮, ▥, ▯, ▬, ▤, or ▭. These settings can also be changed by clicking on the displayed icon.

The default page rendering mode is Paper, which corresponds to the way your document will be printed. However, in this mode, the positions of page breaks may heavily change while you are typing, which can be annoying. This can be avoided by using the alternative Papyrus mode, in which your document is displayed on a single page of fixed width and flexible height. The Papyrus mode also allows for faster typesetting, which makes the editor more responsive when your document gets large.

Font. The main document font and its size are indicated and can be modified from the focus bar. The proposed main fonts are on a restricted shortlist of fonts that are particularly well supported by TeXmacs.

Language. The icon with the little flag indicates the main document language. This language is used by the typesetter for punctuation and hyphenation of words. It is also relevant for the spell checker.

Section. If your document contains several sections, then the focus bar also indicates the current section. By clicking on it, a list with all sections appears, which allows you to jump to any other section inside the document.

2.8 Finding documentation

The easiest way to find documentation on a topic inside TeXmacs is to search for it using `F1` or Help ▸ Search ▸ Documentation. Search queries are similar to those for popular search engines on the web: just type one or more keywords separated by spaces and TeXmacs will return a list of matching pages in the documentation, ordered by expected relevance.

We also recall from the previous subsection that Focus ▸ Describe provides contextual help on the current focus tag. If the focus is on the entire document, then Focus ▸ Help will provide help on the current document style and any selected style packages. Minimal contextual explanations of icons and menu entries are also available via help balloons that appear as soon as you hover your mouse pointer over them for a short while.

The bulk of the built-in documentation on TeXmacs is available from the Help menu. The available material is organized as follows:

- In the Help ▸ Manual menu, you find the main documentation about how to use TeXmacs. The manual is quite complete, but not very detailed.

- In the Help ▸ Reference guide menu, you find more detailed documentation about the TeXmacs format, its primitives and environment variables, and the standard style files.

- TeXmacs plug-ins may (or may not) include documentation. Any available auto-documentation of this kind is made available through the Help ▸ Plug-ins menu.

- The Help ▸ About menu contains more information about TeXmacs, like information about its authors, how to contact us, and changes in different versions of the program.

- More information is available for developers (Help ▸ Interfacing, Help ▸ Scheme extensions, Help ▸ Future plans) and anyone who wants to support the development of TeXmacs (Help ▸ Help us).

In addition, book versions of the existing documentation are available in Help ▸ Full manuals. An interesting feature of TeXmacs is that any documentation page can be expanded into an article or a full manual using Help ▸ Full manuals ▸ Compile article or Help ▸ Full manuals ▸ Compile book. For instance, if you start browsing the manual using Help ▸ Manual ▸ Browse, and click on Help ▸ Full manuals ▸ Compile book, then TeXmacs will compile the user manual [25]. Please be patient… Compiling a full manual may take a few minutes…

The Internet provides another valuable source of information. On the TeXmacs web site `http://www.texmacs.org`, you may find videos, tutorials and research papers about TeXmacs. The FAQ and the mailing list archives also contain answers for many questions by users. Do not hesitate to subscribe to the mailing lists and ask your own questions there, or help others out!

Chapter 3

Ready, steady, go!

3.1 Typing ordinary text

When it comes to typing ordinary text, we have nothing extraordinary to say: most of the time, T_EX_{MACS} will behave similarly to other programs on your computer. Nevertheless, there are a few points worth mentioning for entering certain symbols and for getting the punctuation, spacing, and other typographic details right.

Special symbols

TeXmacs reserves the `\`, `$`, and `"` keys for entering special commands, mathematical formulas, and quotes. The literal symbols \, $, and " can be produced using the shortcuts `⇧F5` `\`, `⇧F5` `$`, and `⇧F5` `"`, or alternatively using `\` `⇥`, `$` `⇥`, and `"` `⇥`. The `⇧F5` keyboard prefix is also used for entering other special symbols; see Table 3.1.

Punctuation

TeXmacs assumes that you are familiar with basic punctuation rules for the language you are writing in. English rules stipulate that each of the principal punctuation symbols

. , : ; ! ?

should be followed by a space, except at the end of a paragraph, but not preceded by any spaces. French rules on the contrary require you to also put a space before these symbols, except for periods and commas.

Assuming that you correctly follow these language-specific rules, TeXmacs takes care of the more precise spacing conventions around punctuation symbols. For example, periods are followed by slightly larger spaces than commas. This makes it easier for the eye to spot separate sentences. Similarly, French typographers use as slightly smaller “thin” space before colons, semicolons, exclamations marks, and question marks.

Shortcuts							
⇧F5 a	æ	⇧F5 A	Æ	⇧F5 a e	æ	⇧F5 A E	Æ
⇧F5 o	ø	⇧F5 O	Ø	⇧F5 o e	œ	⇧F5 O E	Œ
⇧F5 s	ß	⇧F5 S	S				
⇧F5 !	¡	⇧F5 ?	¿	⇧F5 p	§	⇧F5 P	£

Table 3.1. Typing special characters using the ⇧F5 keyboard prefix.

Whitespace	
negative space xy	\ ! ↩
no space xy	
tiny space xy	\ , ↩
small space x y	\ : ↩
thin space x y	\ ; ↩
normal space x y	␣
quad space x y	␣ ⇥ ⇥ or \ q u a d ↩
double quad space x y	\ q q u a d ↩

Table 3.2. Common amounts of whitespace and how to obtain them.

There are a few exceptional cases that require additional care. The period sign is also used inside abbreviations, for example, in which case it should be followed by a normal space (instead of the slightly larger space between sentences). We recommend that you systematically "tag" abbreviations using Insert ▸ Content tag ▸ Abbreviation or the keyboard shortcut ⌥a. This in particular allows TeXmacs to adjust the spacing.

Whitespace

We have seen above that TeXmacs automatically determines the appropriate amount of whitespace to be inserted for each space character in the document. However, TeXmacs does not force you to enter a space after, say, a period. This flexibility is useful whenever periods are used for other purposes as ending sentences (e.g. inside numbers 1.234, abbreviations, names of websites `www.texmacs.org`, etc.).

It is generally a bad idea to insert several spaces in a row, just for the visual effect of obtaining a somewhat larger space. In the case when you need an explicit whitespace of, say, one inch, you should rather use Format ▸ Whitespace ▸ Rigid and enter `1in` at the prompt. Certain amounts of whitespace are very common for tweaking the presentation of text; see Table 3.2 for some useful shortcuts to this effect. For more details about horizontal whitespace, we refer to section 7.5.1.

Shortcuts			
`	‘	'	’
` `	“	' '	”
< ⇥	‹	> ⇥	›
< <	«	> >	»
⇧F5 "	"	, ,	„

Table 3.3. Keyboard shortcuts for specific quotes.

When entering several spaces in a row, the default behavior of TeXmacs is to glue them together into a single big space. More precisely, hitting *n* times ␣ will produce a space with the width of $n-1$ “quad” spaces. Using Edit ▸ Preferences ▸ Space bar in text mode ▸ Allow multiple spaces, it is also possible to mimic the behavior of most other text editors: hitting *n* times ␣ will simply insert *n* spaces. Similarly, Space bar in text mode ▸ No multiple spaces forbids multiple spaces altogether: pressing ␣ any number of times always inserts a single space.

Hyphens and dashes

TeXmacs provides three types of hyphens and dashes (namely -, –, and —), which are entered using the shortcuts -, - -, and - - -. The shortest ones, called *hyphens*, are used for hyphenation and in order to join words (e.g. high-school teacher, Marie Skłodowska-Curie, etc.). So-called *en dashes* (–) are typically used for ranges (see pages 256–65536), relations, and connections (the Stone–Weierstrass theorem). The longest *em dashes* (—) mostly serve as substitutes for parentheses or colons (Archimedes—the great scientist—is also known for letting his bath overflow). TeXmacs finally provides a *non-breaking hyphen* after which line breaks are forbidden; you may enter this character using - ⇥.

Quotes

When you press the " key, an appropriate opening or closing quote will be inserted. The quote character is chosen according to the current language and the surrounding text. If the default quoting style is not appropriate for some reason, then you can change it in Edit ▸ Preferences ▸ Keyboard ▸ Automatic quotes. You can also insert specific quotes using the keyboard shortcuts in Table 3.3.

In most languages, opening quotes are preceded but not followed by a space, whereas closing quotes are followed but not preceded by one. The French use a thin space after opening *guillemets* « and before closing ones ». The spacing may need to be adjusted for nested quotations, such as “Archimedes exclaimed ‘Eureka!’ ”. In order to detach the single nested closing quote from the surrounding double closing quote, we manually inserted a small space using \ ; ↩.

Font	‘ ’	‘ ‘ ’ ’	‘ ⇥	’ ⇥	" ⇥
TeXmacs Computer Modern	‘ ’	“ ”	‘	’	"
TeX Gyre Termes (Times)	‘ ’	“ ”	`	'	"
TeX Gyre Pagella (Palatino)	‘ ’	“ ”	`	'	"
DejaVu	‘ ’	“ ”	`	'	"

Table 3.4. Rendering of simple quotes, double quotes, backquotes, apostrophes, and quotation marks in various fonts.

The single quotes ‘ ’ and the double closing quotes ” are easily confused with the backquote `, the ambidextrous apostrophe ' and the ambidextrous quotation mark ". Older fonts may even display some of these symbols in the same way, whereas modern Unicode fonts tend to propose different glyphs (see Table 3.4). “Genuine” single and double quotes are used for quotations in ordinary text; they can for instance be entered using the natural shortcuts ‘, ’, ‘ ‘, and ’ ’, as explained above. The quotation mark ", the apostrophe ', and the backquote ` are important for programmers, who tend to use notations like `"Hello world"` or `’Hello world’` for representing textual string data; you may enter them using ‘ ⇥, ’ ⇥, and " ⇥. Notice that the keyboard shortcuts are “reversed” inside computer programs (e.g. " produces the quotation mark ", whereas " ⇥ produces ”).

Line breaking

Ordinary text is built up from paragraphs, each of which contain one or more sentences. Most paragraphs do not fit on a single line, so TeXmacs automatically breaks them into several lines. Sometimes, you may wish to control more precisely how this is done. There are several circumstances in which line breaks should be avoided:

- After initials of names: do not break after “A.” in “A. Einstein”.
- After very short words: I personally tend to avoid breaks after single-letter words such as the article “a”.
- Before references such as the number “3” in “Chapter 3”.
- Inside mathematical text, it is often considered bad taste to start a line with a mathematical formula.

TeXmacs provides a special “non-breaking space” character that can be used instead of an ordinary space in order to prevent line breaks at a specific position. This character is entered using ␣ ⇥ and rendered on the screen using a barely visible steel-blue hyphen. You may also prevent all line breaks inside a selection of text by turning it into a “group” using Format ▸ Break ▸ Group ▸ Horizontal or ⌘=.

Conversely, you may insist on having a line break at a certain point using Format ▸ Line break. Notice that it is generally better to forbid breaks at places that are inappropriate for sure rather than to force them at places that look nice at a time *t*. Indeed, as soon as you make further modifications inside your text, change the document style, or export it to another format, the nice looking break often becomes your worst possible choice.

Hyphenation

Long words may be hyphenated during the line breaking process. The hyphenation rules are determined by the language in which you wrote your text (as selected using Document ▸ Language or Insert ▸ Language).

Again, it is sometimes desirable to have more control over the hyphenation process. Now the above mechanism of "grouping" does not only forbid line breaks: it also prevents any words from being hyphenated. It is typically a good idea to turn off hyphenation inside names of persons (try hyphenating your own name).

For a given word, TeXmacs also allows you to explicitly specify its hyphenation: select the word and use Format ▸ Hyphenate as. You will be prompted for a hyphenation of the word. Just enter short dashes at the places where hyphenations should be allowed.

3.2 Foreign languages

In the past, typing foreign text on a computer was often a complex task, especially for languages with other alphabets or scripts, such as Russian or Chinese. The UNICODE standard was designed to simplify this task and provide a "universal" computer alphabet for all known languages in the world. Nowadays, most operating systems comply with this standard and provide built-in mechanisms for entering text in UNICODE format, while supporting a large number of languages and keyboard models.

TeXmacs is also compatible with a large part of UNICODE (notable exceptions are Arab and Hebrew which are written from right to left), even though another encoding is used internally. For the increasingly rare cases in which it is impossible or difficult to enter certain characters using the built-in facilities of the operating system, TeXmacs provides a few additional mechanisms of its own, that we will describe below.

Note that you can set the text language for the whole document with Document ▸ Language. It is also possible to select another language for a single word or fragment of text using Insert ▸ Language. TeXmacs uses language information for a variety of purposes, such as hyphenation, spell checking, available keyboard shortcuts, punctuation and spacing rules, etc. It is also possible to specify an alternative language for the graphical user interface in Edit ▸ Preferences ▸ General ▸ Language.

Shortcut		Example		Shortcut		Example	
⌘'	Acute ´	⌘' e	é	⌘`	Grave `	⌘` e	è
⌘^	Hat ˆ	⌘^ e	ê	⌘"	Umlaut ¨	⌘" e	ë
⌘~	Tilde ˜	⌘~ a	ã	⌘C	Cedilla ¸	⌘C c	ç
⌘U	Breve ˘	⌘U g	ğ	⌘V	Check ˇ	⌘V s	š
⌘O	Above ring ˚	⌘O a	å	⌘.	Above dot ˙	⌘. z	ż
⌘H	Hungarian ˝	⌘H o	ő				

Table 3.5. Typing accented characters.

3.2.1 Latin languages

Among the languages with Latin alphabets, the English language is particularly simple in the sense that its letters do not carry any accents. For other "Latin" languages, you may enter accented letters using the ⌘ prefix. For example, "é" is obtained by typing ⌘' e, and "à" by typing ⌘` a. See Table 3.5 for the complete list of available accents, with the corresponding keyboard shortcuts and examples. Note that these shortcuts are available regardless of the current language setting.

In Table 3.1 we already showed how to enter several other special letters and symbols using the prefix ⇧F5. Further keyboard shortcuts are available for specific languages only: see Table 3.6. Some of these shortcuts override general shortcuts from Table 3.1. Using the "variant" mechanism (see section 2.3.2), it is possible to recover the overridden behavior. For example, inside Hungarian text, you can still obtain "ø" through the shortcut ⇧F5 o ⇥.

3.3 Starting a new document

In TeXmacs, the first thing that you usually do when starting a new document is to choose an appropriate *document style*. This can be done using Document ▸ Style or directly via the focus bar.

Hungarian	
⇧F5 o	ő
⇧F5 O	Ő
⇧F5 u	ű
⇧F5 U	Ű

Spanish	
! ⇥	¡
? ⇥	¿
! `	¡
? `	¿

Polish			
⇧F5 a	ą	⇧F5 o	ó
⇧F5 A	Ą	⇧F5 O	Ó
⇧F5 c	ć	⇧F5 s	ś
⇧F5 C	Ć	⇧F5 S	Ś
⇧F5 e	ę	⇧F5 x	ź
⇧F5 E	Ę	⇧F5 X	Ź
⇧F5 l	ł	⇧F5 z	ż
⇧F5 L	Ł	⇧F5 Z	Ż
⇧F5 n	ń	⇧F5 z ⇥	ź
⇧F5 N	Ń	⇧F5 Z ⇥	Ź

Table 3.6. Language-specific text shorthands.

TeXmacs includes a few general purpose styles such as article, beamer, book, Education ▸ exam and letter, as well as a selection of more specialized styles. For example, in the menus Article and Book, you will find additional styles for writing articles and books, implementing specific layouts that are required by certain journals or publishers.

Assuming that you selected a document style such as “article” or “book”, the next logical actions are to enter a title, the authors, an abstract, etc. Of course, the precise kind of information you provide depends on the style: in the case of an exam, you may wish to add the subject, the level, or the name of a classroom; the title page of a laptop presentation typically becomes more attractive if you add some artwork.

3.3.1 Entering the title

Let us detail how to enter title information in the case of an article. Such information should be regarded as a block of data. The specific order and way in which the data are presented is determined by the document style, although TeXmacs provides a few options for customization.

You may use Insert ▸ Title ▸ Insert title in order to create a block with title information. For a newly created document, you may also use the Title button on the focus bar. The main data field of the title block is the title itself. Extra data fields can be inserted from the Focus menu or the + icon on the focus toolbar. For instance, Focus ▸ Author allows you to insert one or more authors. You may also specify the creation date of your document using Focus ▸ Date (arbitrary dates) or Focus ▸ Today (the current date). Miscellaneous fields with a customized layout can be added using Focus ▸ Miscellaneous. For notes with people to be thanked or grant information, you should rather use Focus ▸ Note.

Many document styles do not only display the title at the top of the first page: often this information also shows up in the headers of individual pages. If a title is very long, then this may cause the headers to run off the pages. In such cases, you may specify an alternative *running title* using Focus ▸ Running title. The same remark applies to the authors: if your paper has many authors, then you should specify an abbreviated *running author* for page headers using Focus ▸ Running author.

For each author that you enter using Focus ▸ Author, you provide additional data via the items below Focus ▸ Author (alternatively you may use the + icon next to Author on the focus bar). Nowadays it is customary to provide an affiliation for the author (Focus ▸ Author ▸ Affiliation) as well as an email address (Focus ▸ Author ▸ Email). Optionally, you may also wish to point to a homepage (Focus ▸ Author ▸ Homepage).

Miscellaneous data fields and further notes can be inserted in a similar way as for titles.

Optimale Pareto-efficiëntie in genotssamenlevingen[1]

JAN KLAASSEN
Afdeling wiskunde
Katholieke Universiteit Tilburg

Email: `jan@wis.kunt.nl`

ADA P. NOOT-MIES
Afdeling econometrie
Universiteit van Amsterdam

Email: `ada@uva.nl`

PIET SNOT
Afdeling wiskunde
Katholieke Universiteit Tilburg

Email: `piet@wis.kunt.nl`

Optimale Pareto-efficiëntie in genotssamenlevingen[2]

JAN KLAASSEN[a], PIET SNOT[b]
Afdeling wiskunde
Katholieke Universiteit Tilburg

a. Email: `jan@wis.kunt.nl`
b. Email: `piet@wis.kunt.nl`

ADA P. NOOT-MIES
Afdeling econometrie
Universiteit van Amsterdam

Email: `ada@uva.nl`

Optimale Pareto-efficiëntie in genotssamenlevingen[3]

JAN KLAASSEN[ab], ADA P. NOOT-MIES[cd], PIET SNOT[ae]

a. Afdeling wiskunde
Katholieke Universiteit Tilburg

c. Afdeling econometrie
Universiteit van Amsterdam

b. Email: `jan@wis.kunt.nl`
d. Email: `ada@uva.nl`
e. Email: `piet@wis.kunt.nl`

Figure 3.1. Three ways to cluster title information: [1] none; [2] by affiliation; [3] maximal.

Notice that data fields need not be unique. You may for example specify more than one affiliation or email address for an author, or attach several miscellaneous data fields to the title.

Whenever several authors share the same affiliation, publishers may require you to save space by factoring out such common information. TeXmacs can do this automatically for you; the desired amount of "clustering" can be specified using the following options in the 🔧 menu on the focus bar:

No clustering. Do not factor out any data.

Cluster by affiliation. Group authors by their affiliations.

Maximal clustering. Factor out as much common data as possible.

The effect of these options is illustrated in Figure 3.1.

3.3.2 Entering the abstract

The abstract of an article can be entered using Insert ▸ Title ▸ Abstract. This action really inserts *abstract blocks* which are similar to the title blocks from the previous section: besides the abstract itself, you may also insert several metadata such as keywords and subject classifiers.

More precisely, with the current focus on the abstract, a list of keywords can be entered using Focus ▸ Keywords: you start typing the first keyword and then insert additional keywords using Focus ▸ Insert argument after or ⌥→. Each topic (mathematics, physics, computer science, etc.) admits its own specific subject classification system. For example, mathematicians often use the A.M.S. subject classification (Focus ▸ A.M.S. subject class), whereas computer scientists tend to prefer the A.C.M. computing class (Focus ▸ A.C.M. computing class). In TeXmacs, we aim at supporting the most frequent classification schemes, and new schemes can easily be added on request.

We notice that certain metadata can also be entered using Document ▸ Metadata. This alternative mechanism attaches the metadata even more directly to the document. Currently, you may specify a title, an author, and a subject in this way, while TeXmacs attempts to automatically determine as many fields as possible. This kind of metadata is for instance used when exporting a document as a PDF file.

3.4 Subdividing your document into sections

Long documents are usually subdivided into chapters and/or sections (laptop and slide presentations rather consist of a sequence of "screens" or slides). As a structured text editor (see section 1.3), TeXmacs provides special markup for sections. We recommend to use this markup and to refrain from manually typesetting section titles using, say, a large underlined font.

New sections can be inserted using Insert ▸ Section or ⌥1. There are different levels of sections. At the top level, books are first subdivided into parts and chapters. Articles and chapters of books are next subdivided into sections, subsections and subsubsections. Smaller portions of text can finally be put into paragraphs and subparagraphs. The level of a section can be changed *a posteriori* using the variant system. For instance, you may use ^⇥ in order to change a section into subsection or a subsubsection into a paragraph.

Sections are numbered by default. You may toggle the numbering of individual sections using ^# or the Ⅳ icon on the focus toolbar. Paragraphs and subparagraphs form an exception: in most of the standard document styles they are not numbered , but this default behavior can be overridden via Focus ▸ Preferences ▸ Paragraph display numbers.

TeXmacs also provides an appendix tag for appendices at the end of a paper or book. Appendices are numbered A, B, C, … instead of 1, 2, 3, … In a similar vein, books may include a prologue (Insert ▸ Chapter ▸ Prologue) or epilogue (Insert ▸ Chapter ▸ Epilogue).

There are a few special types of unnumbered sections (or chapters) for which the content can be generated automatically: tables of contents, bibliographies, indexes, glossaries, lists of figures, etc. The mechanisms for automatic content generation will be explained in section 6.4.

The subdivision of your document into sections provides a skeleton that is useful for several purposes. We already mentioned the generation of tables of contents. We also recall that the focus bar displays the current section title, whenever the focus is on the entire document. By clicking on this title on the focus bar, TeXmacs opens a menu that allows you to easily jump to any of the other sections. When exporting a document to PDF, a suitable table of contents is also automatically included.

3.5 Quotations and prominent statements

There are many occasions on which a fragment of text needs to catch the eye. Mathematicians like to state their theorems prominently, whereas literary writings may use a special layout for quotations or poetry. High school teachers emphasize the most crucial parts of their courses inside frames. Being a structured editor, TeXmacs encourages you to use special markup in each of these situations. This in particular guarantees a uniform layout.

The markup described in this section is mainly intended for text fragments of medium length. In LaTeX, such markup elements are called *environments*. People with HTML [41] or XML [3] backgrounds prefer the terminology *block markup*. In section 3.7 below, we focus on *inline markup* for shorter pieces of text, such as emphasized definitions of words or monospaced function names inside computer programs.

3.5.1 Theorems

The backbone of mathematical discourse consists of theorems, proofs, and other auxiliary propositions and lemmas. Numerous documents in other areas are structured likewise, along hypotheses, principles, remarks, warnings, etc. In TeXmacs, such structures are called *enunciations*, and you may insert them using Insert ▸ Enunciation.

The most important "theorem-like" enunciations are emphasized, whereas the more casual "remark-like" enunciations are not. TeXmacs also includes several "exercise-like" enunciations for educational purposes, which may use a slightly different layout. Here follows an example of each of these three main types of enunciations:

THEOREM 3.1. *On a Turing machine with sufficiently many tapes, two n-digit numbers can be multiplied in time* $O(n \log n)$.

Note 3.2. The history of multiplication algorithms is a fascinating one, which goes back to ancient Babylon. The above theorem from [23] corresponds to the best currently known algorithm on a sequential computer.

> **Exercise 3.1.** Show that the Euclidean division of a $2n$-digit integer by an n-digit one can also be computed in time $O(n \log n)$.

It is nice to give credits in the very statement of a theorem. From within an enunciation, this can be done using Focus ▸ Due to or the Due to button on the focus bar.

By default, enunciations are numbered. Assuming that your cursor is inside an enunciation, you may remove its number using the toggle Focus ▸ Numbered or the icon Ⅳ on the focus toolbar.

Americans use the convention that theorems, lemmas, propositions, remarks, etc. all share the same *counter*. This explains why the above Note is numbered 3.2, even though it is the first Note in this chapter. Some people prefer the "European numbering style", where each kind of enunciation carries its own individual counter. You may select this alternative numbering style using Focus ▸ Preferences ▸ European numbering style.

In books, TeXmacs prefixes the numbers of enunciations by the chapter number. For long articles or books with long chapters, you may prefer to use the section number as the prefix. This can be done using Focus ▸ Preferences ▸ Prefix by section number.

The Focus ▸ Preferences menu of an enunciation contains a few other interesting items that you may wish to try in order to customize the rendering.

3.5.2 Quotations and poetry

Three environments for quotations and poetry have been adapted from LATEX to TEXMACS. The first one, quote (Insert ▸ Prominent ▸ Quote), should be used for one or more short, one-line, quotations.

Eureka! — *Archimedes*

$E=mc^2$ — *Einstein*

For longer quotations of more than one paragraph, you should rather use quotation (Insert ▸ Prominent ▸ Quotation).

For poetry, you may use the verse tag (Insert ▸ Prominent ▸ Verse). The layout is similar to quotations, but differs when it comes to line breaking. This is best seen when making the paragraph artificially narrow:

Het was in eenen tsinxen
daghe
Dat beede bosch ende haghe
Met groenen loveren waren
bevaen.
Nobel, die coninc, hadde ghe-
daen
Sijn hof crayeren over al
Dat hi waende, hadde hijs
gheval,
Houden ten wel groeten love.
Doe quamen tes sconinx hove
Alle die diere, groet ende
cleene,
Sonder vos Reynaert alleene.
Hi hadde te hove so vele mes-
daen
⋮

Van den vos Reynaerde

3.5.3 Other prominent text

The Insert ▸ Prominent menu also contains various items to single out portions of text in a visually oriented way. First of all, you may change the alignment of a block of text using Insert ▸ Prominent ▸ Left aligned, Insert ▸ Prominent ▸ Centered and Insert ▸ Prominent ▸ Right aligned:

Left aligned

Centered

Right aligned

TeXmacs next provides a series of tags that allow you to surround a block of text by vertical padding, horizontal lines, or a frame. These are illustrated below:

> Use Insert ▸ Prominent ▸ Padded to insert vertical padding around text.

> You may also put horizontal lines around it using Insert ▸ Prominent ▸ Lines around. If you only want a line above or below, then you should use Insert ▸ Prominent ▸ Overlined or Insert ▸ Prominent ▸ Underlined.

> Really important statements can be framed using Insert ▸ Prominent ▸ Framed.

These tags are often used in conjunction with other ones. For instance, we used quotations for each of the above examples in order to make clear where they start and end.

The markup for padding, overlines, underlines, and frames comes with several style options. The precise amount of padding can be changed via Focus ▸ Above and Focus ▸ Below or directly on the focus toolbar. The default amount of padding for all similar padded structures in your document can be specified using Focus ▸ Preferences ▸ Padding above and Focus ▸ Preferences ▸ Padding below. For overlined and underlined structures, you may also change the amount of space that separates the text from the lines. Framed boxes allow you to specify a similar horizontal separation space, as well as a background color.

Ornaments (Insert ▸ Prominent ▸ Ornament) are a more general kind of frames with even more bells and whistles. They are most convenient for laptop presentations, where you may use them for obtaining various graphical effects:

> THEOREM 3.3. *This is a theorem to remember.*

3.6 Item lists

TeXmacs implements three main types of lists, depending on whether the items are numbered or not and whether they contain a brief description.

Using Insert ▸ Itemize you may start a list with unnumbered items. You may optionally select a specific item tag like • (bullets), – (dashes), or → (arrows), to be put in front if the items in the list. Lists may be *nested* inside other lists, as follows:

- First item.
- Now comes the sublist:
  - ∘ A subitem.
  - ∘ Another one.
- A final item.

The rendering of the default item tag depends on the level of nesting: the • item tag for the outermost level, the ∘ item tag at the second level, and so on.

Numbered lists can be inserted using Insert ▸ Enumerate. Again, you may optionally choose a specific numbering style. Here is an example of an enumeration that was started using Insert ▸ Enumerate ▸ Roman:

I. A first item.

II. A second one.

III. And a last one.

The default numbering style again depends on the nesting level: at the outermost level, most styles use ordinary numbers. At the second level, it is common to use lowercase letters, and so on.

Descriptive lists are started using Insert ▸ Description and they allow you to describe a series of concepts:

Gnu. A hairy but gentle beast.

Gnat. Only lives in a zoo.

Yet once more, TeXmacs proposes several presentation styles, some of which are only appropriate if you know beforehand that the keys Gnu, Gnat, etc. are sufficiently short.

Assuming that you are inside an item list, pressing ↩ automatically starts a new item. If you need items that are several paragraphs long, then you can use ⇧↩ in order to start a new paragraph. TeXmacs also allows you to turn an unnumbered item list into a numbered one and *vice versa*, using the toggle Focus ▸ Numbered or the icon Ⅳ on the focus toolbar. The default layout of lists is rather spaced out. You may select a more compact rendering style using Focus ▸ Preferences ▸ Compact Lists. The low-level style parameters that control the spacing of icons can also be customized via the style preferences menu Focus ▸ Preferences.

Menu	Tag	Example	Purpose
Strong	strong	this is **important**	Indicate an important region of text
Emphasize	em	the *real* thing	Emphasize a region of text
Definition	dfn	A *gnu* is a horny beast	Definition of some concept
Sample	samp	the ae ligature æ	A sequence of literal characters
Name	name	the LINUX system	The name of a particular thing
Person	person	NIELS, il touche !	The name of a person
Cite	cite*	Melville's *Moby Dick*	A bibliographic citation
Abbreviation	abbr	I work at the C.N.R.S.	An abbreviation
Acronym	acronym	the HTML format	An acronym
Verbatim	verbatim	the program said `hello`	Verbatim text like program output
Keyboard	kbd	Please type `return`	Text to be entered on a keyboard
Code	code*	`cout << 1+1;` yields 2	Code of a computer program
Variable	var	`cp` *`srcfile destfile`*	Variables in a computer program
Deleted	deleted	Remove the ~~t~~i~~p~~o	Mark text that has been deleted
Fill out	fill-out	My name is Sylvie	Text to be filled out
Marked	marked	Remind feeding the cat	A crucial passage

Table 3.7. Common content-based tags and how to obtain them via Insert ▸ Content tag.

3.7 Special textual markup

We have already encountered various *block markup* elements for prominent text of one or more paragraphs (see section 3.5). Similar *inline markup* elements for shorter pieces of text can be found in the submenu Insert ▸ Content tag.

For example, using Insert ▸ Content tag ▸ Strong, you can mark important words or concepts. The default rendering of **important text** uses a bold typeface, but certain document styles may opt for an alternative presentation. For this reason, you should *not* use the strong tag as a substitute for "bold rendering", but only for the precise purpose of singling out important fragments of text.

We have listed the most common inline content tags in Table 3.7. Due to name clashes, some tags do not carry the most intuitive names: the block tag definition is one of the standard TeXmacs enunciations (see section 3.5.1), the tag cite for citations based on bibliographic databases (see section 6.5), and code for multiple lines of code.

In addition to the above inline content markup, TeXmacs also provides several kinds of presentation markup. First of all, tags for several standard font sizes can be found in Insert ▸ Size tag. These sizes are relative to the size of the main document font. For instance, assuming that the main document is composed using a 10 point font, "small" text will use an 8 point font; if the base font size is 12 point, then "small" text rather corresponds to a 10 point font. See Table 3.8 for the available size tags. Notice that you may use ^⇥ and ^⇧⇥ to cycle among the various font sizes.

Size tag	Example
really-tiny	Really tiny
tiny	Tiny
very-small	Very small
small	Small
normal-size	Normal

Size tag	Example
large	Large
very-large	Very large
huge	Huge
really-huge	Really huge

Table 3.8. Common size tags available through Insert ▸ Size tag.

Other presentation markup can be found in Insert ▸ Presentation tag: you may use Underline, Overline, and Strike through for underlined, overlined, and ~~crossed out~~ text, respectively. (Although underlining is a common way to emphasize text in handwritten documents, we notice that this practice is considered bad taste for printed material.) Subscripts and superscripts as in 2nd can be obtained using Subscript and Superscript or the keyboard shortcuts _ ⇥ and ^ ⇥. Using Pastel, Greyed, and Light, you may create translucent text with various degrees of transparency. This feature is typically used for laptop presentations in order to anticipate upcoming content. The graphical effect is best noticed when using colored text on background patterns.

Pastel red, green, and blue text
Greyed red, green, and blue text
Light red, green, and blue text
Normal red, green, and blue text

3.8 Fonts

As we have discussed at length by now (see sections 1.3 and 3.7), traditional typesetters tend to use special fonts only for specific, well-identified purposes. Users are recommended to use the appropriate markup for these purposes, rather than changing the font explicitly. Nevertheless, you may sometimes wish to select a particular font yourself, or search for a font with certain characteristics (e.g. a bold handwritten font).

3.8.1 Font management

Before you select a nominal font, beware of the fact that high quality fonts are really collections of fonts. It is quite common that a font comes with bold and italic variants. Rich font collections may implement even more variants: sans serif, `monospaced`, SMALL CAPITALS, etc. In addition, modern UNICODE fonts do not only provide glyphs for English text, but also for accented characters and other scripts (Greek, Cyrillic, Chinese, etc.). Similarly, scientific documents with mathematical formulas may require a UNICODE font with a large set of mathematical symbols [64, 2].

In comparison with other non-structured text editors, this idea of *font collections* is particularly important in TEXMACS. Indeed, assume that you have selected one of your favorite fonts. Which font is TEXMACS supposed to use when you start a mathematical formula, when a monospaced variant is needed for typesetting a computer program, or when part of your document is written in an exotic language? There are only a very limited number of fonts that allow you to do all this. Examples include the Computer Modern [37] and Stix fonts [54], as well as the TEX Gyre fonts Bonum, Pagella, Schola, Termes, and DejaVu [32].

The above discussion actually raises three main questions. First of all, how to find those font collections that support a certain number of features (e.g. all Cyrillic fonts with bold and italic variants)? Secondly, what to do if a certain feature is not available for the user's favorite font (e.g. typeset a computer program using a font with no monospaced variant)? Finally, how to specify separate fonts for specific tasks (one font for normal text, another one for bold section titles, and yet another one for mathematical formulas)?

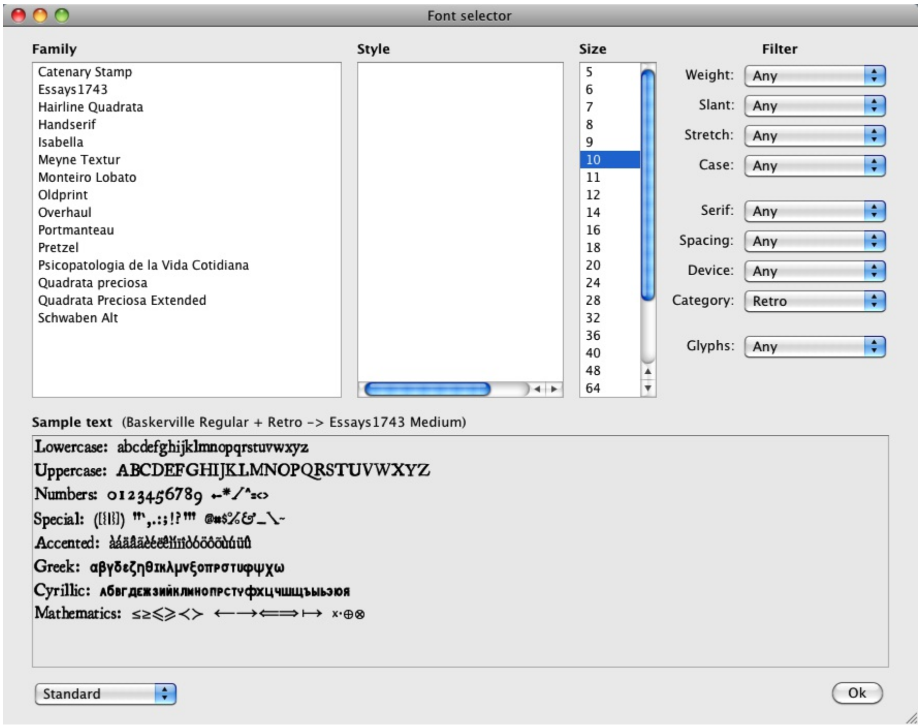
TEXMACS partially addresses these problems by maintaining a detailed *font database*. This database contains more information about fonts than the fonts themselves. For instance, TEXMACS is aware which fonts are sans serif, handwritten, outlined, monospaced, etc. The database also contains more subtle technical information about the general shape of the font (its boldness, aspect ratio, slant, ascent, descent, etc.). In the case when a font with certain required properties does not exist, this allows TEXMACS to search for the best available proxy. The same mechanism applies when a particular character is missing in a font.

The downside to the above approach is that part of the information in the database needs to be entered manually. Although the database contains entries for most of the standard fonts available under GNU/LINUX, MACOS, and WINDOWS, it may not recognize any further fonts that you may have purchased or installed on your system.

Another thing to keep in mind when using less common fonts is that TEXMACS comes with only a limited number of preinstalled fonts, such as the Stix fonts, the TeX Gyre fonts, and several fonts prefixed by "TeXmacs". Documents that only use these fonts will be rendered in the same way on different systems (assuming the versions of TEXMACS are identical). Any other fonts may be replaced by closest available proxies according to the font database.

3.8.2 Selecting a font

In TEXMACS, the global document font can be specified using Document ▸ Font. It is also possible to locally use another font using Format ▸ Font. Both Document ▸ Font and Format ▸ Font open the TEXMACS font browser, which looks like the window in Figure 3.2. The standard way to indicate fonts is via their Family, Style and Size (in points). In addition, TEXMACS provides several filters that allow you to search for fonts that meet one or more criteria.



Catenary Stamp
Essays1743
Hairline Quadrata
Handserif
Isabella
Meyne Textur
Monteiro Lobato
Oldprint
Overhaul
Portmanteau
Pretzel
Psicopatologia de la Vida Cotidiana
Quadrata preciosa
Quadrata Preciosa Extended
Schwaben Alt

Size: 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 24, 28, 32, 36, 40, 48, 64 (10 selected)

Weight:	Any
Slant:	Any
Stretch:	Any
Case:	Any
Serif:	Any
Spacing:	Any
Device:	Any
Category:	Retro
Glyphs:	Any

Sample text (Baskerville Regular + Retro -> Essays1743 Medium)

Lowercase: abcdefghijklmnopqrstuvwxyz
Uppercase: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Numbers: 0123456789 +-*/^=<>
Special: ([{|}]) "`,.:;!?"' @#$%&_\~
Accented: àáâãäåèéêìíòóôõöùúûü
Greek: αβγδεζηθικλμνξοπρστυφψχω
Cyrillic: абвгдежзийклмнопрстуфхцчшщъыьэюя
Mathematics: ≤≥≶≷<> ←→⟸⟹↦ ×·⊕⊗

Figure 3.2. The TEXMACS font selector.

In the example, we first selected the "Baskerville Regular" font. Using the Category pop-up choice list on the right, we next searched for a font with a "Retro" look and feel that combines as well as possible with the Baskerville font. The header of the Sample text indicates that "Essays1743 Medium" is the best match that we could find on our system. The standard sample text shows the rendering of various characters in this font (other sample texts can be selected instead from the choice list that indicates Standard). The Family frame lists all "Retro" fonts that were found on our system. Since we did not select any of these font families yet, the Style frame does not contain any corresponding styles.

The easiest way to select a font is by clicking on a suitable family and then press Ok. If you rather select your font by picking a style or by using a filter (as in our example), then the semantics of font selection is slightly more subtle. For instance, when clicking on Ok in Figure 3.2, TEXMACS will insert a tag inside the document that asks for a "Retro" font. Since the surrounding text was typeset using the "Baskerville Regular" font, any text that you will enter next will be typeset using the "Essays1743 Medium" font, as indicated by the font selector; so far so good. However, whenever you copy and paste the newly entered text somewhere else, the appropriate "Retro" font for the surrounding text is not necessarily "Essays1743 Medium": TEXMACS might use another, more relevant font instead.

Although the font selector is easy to use, it is more limited when it comes to specifying a more precise fallback strategy in the case of missing fonts or characters. For example, it does not allow you to specify one particular font for text and another one for mathematics. $\TeX_{\text{MACS}}$ provides a mechanism for building composite of this type, which is beyond the scope of this book, and currently only intended for expert users (if you are curious, then you may try Edit ▸ Preferences ▸ Other ▸ Advanced font customization before opening the font selector). Another way to force a particular font for, say, mathematics is to adapt the math macro so as to use this font.

That said, we notice that $\TeX_{\text{MACS}}$ is good at emulating mathematical symbols for fonts that do not natively support them. For example, in case of the **Essays1743** font, the arrow → is obtained by juxtaposing two minus signs - and the closing guillemet ›; we refer to [27] for more details.

3.8.3 Font characteristics

As explained above, fonts have three main characteristics:

Family. Fonts are grouped together into *families* with a similar design.

Shape. Inside the same font family, individual fonts have different *shapes*, such as bold, italic, small capitals, etc.

Size. The font *size* in points.

The actual font files on your hard disk usually correspond to the font family and the font shape; the font itself can automatically be scaled to any size.

The $\TeX_{\text{MACS}}$ font database keeps track of several extra characteristics of fonts. The font selector allows you to filter on the following criteria:

Weight. The font *weight* corresponds to the "thickness" of the font:

Thin	Light	Medium	**Bold**	**Black**

Slant. The font *slant* determines the angle of the font:

Normal	*Italic*	*Oblique*

Stretch. This property determines the horizontal width for a fixed vertical height:

Condensed	Unextended	Wide

Case. This property determines how lowercase letters are capitalized:

Mixed	SMALL CAPITALS

Serif. This feature corresponds to decorations at the end of strokes that remind of letters carved into stone:

Serif	Sans Serif

Spacing. This feature corresponds to the horizontal spacing between characters:

Proportional	Monospaced

Device. This property can be used to imitate specific "writing devices":

Print	Typewriter	Digital	Pen	Art pen	Chalk	Marker

Category. Various other font features:

Ancient	Attached	Calligraphic	Comic
Decorative	Distorted	Gothic	Handwritten
INITIALS	Medieval	Miscellaneous	Outline
Retro	SciFi	Title	

Each of the above properties really constitutes a *hint* on the kind of font that *should* be used. If no suitable font can be found on your particular system, then setting these properties may have no effect.

Let us finally recall that the left footer of the $\mathrm{T_EX_{MACS}}$ window indicates some of the most significant properties of the font being used at the current cursor position. For instance, for a generic newly created document, it displays "text roman 10", which means that you are working in text mode using Knuth's 10 point Computer Modern "Roman" font [37].

3.9 Paragraph layout

Most of the time, the layout of paragraphs is determined automatically, as a function of the document style and structured markup inside your text. You may sometimes want to adjust this default layout. This can either be done globally, using Document ▸ Paragraph, or for one or more specific paragraphs, using Format ▸ Paragraph. We recall from section 3.5 that specific rendering effects can also be obtained using the block markup elements in Insert ▸ Prominent.

The alignment of paragraphs can be changed in the Format ▸ Paragraph ▸ Basic ▸ Alignment choice list. The four possible alignment styles are Left, Centered, Right and Justified. The default style is Justified, meaning that full lines are stretched as much as needed in order to align their right-hand sides.

There are two classical ways to make the beginnings of successive paragraphs easy to spot for the reader (see Figure 3.3): by indenting the first line of each paragraph or by inserting additional vertical whitespace between consecutive paragraphs. The extra indentation to be used for first lines of paragraphs can be specified in Format ▸ Paragraph ▸ Basic ▸ First indentation. The amount of whitespace between successive paragraphs can be changed in Format ▸ Paragraph ▸ Basic ▸ Interparagraph space.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox again jumps over the lazy dog. The quick brown fox again jumps over the lazy dog.

The quick brown fox is getting tired, jumping over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox again jumps over the lazy dog. The quick brown fox again jumps over the lazy dog.

The quick brown fox is getting tired, jumping over the lazy dog.

Figure 3.3. Two common paragraph styles.

At this point it is useful to say a few words about *lengths* in TeXmacs. Most of the fields in the Format ▸ Paragraph dialogue window expect length values. For instance, in order to indent every first line of a paragraph by a quarter of an inch, you should enter `0.25in` as the value for Format ▸ Paragraph ▸ Basic ▸ First indentation. A TeXmacs length consists of a number (e.g. `0.25`) followed by a unit (e.g. `in`). Apart from the standard units `mm`, `cm`, and `in`, TeXmacs also provides several units of interest for typographers, such as `pt` (one point; approximately $^1/_{72}$ inch), `fn` (the current font size; e.g. `10pt`), or `tab` (a special unit for indentations; by default, `1tab` equals `1.5fn` and the article style uses a first indentation of `1tab`); see Table 3.9 for an overview.

The dialogue window Format ▸ Paragraph allows you to specify a few other layout properties of paragraphs. The margins can be changed using Left margin and Right margin. For instance, inside quotations (see section 3.5.2), both the left and right margins are increased by `2tab`. The spacing between individual lines is controlled via Interline space. By default, TeXmacs uses a small interline space of `0.2fn`. A zero interline space makes ordinary text less readable, but can be acceptable for computer programs, while stuffing as much code

Unit	Purpose
`cm`	Centimeter
`mm`	Millimeter
`in`	Inch
`pt`	Typographic point
`dd`	Didot point
`pc`	Pica (`12pt`)
`cc`	Cicero (`12dd`)

Unit	Purpose
`fn`	Font size (quad space)
`tab`	Tab space
`spc`	Space in current font
`sep`	Separation space
`ex`	Height of x character
`ln`	Fraction bar width
`par`	Paragraph width
`pag`	Page height

Table 3.9. Available length units in TeXmacs.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

Figure 3.4. Illustration of three different interline spaces: `0fn`, `0.2fn`, and `1fn`.

as possible on the screen. An interline space of `1fn` can be useful for drafts in which readers are likely to make extensive annotations between the lines. These three different settings are illustrated in Figure 3.4.

TeXmacs finally allows you to typeset text in more than one column, while specifying the horizontal space between separate columns (using Number of columns and Column separation). The Format ▸ Paragraph ▸ Advanced tab contains yet more expert settings for the layout of paragraphs.

If your aim is not so much to force a precise layout, but rather to obtain certain graphical effects, then you may prefer to use the markup from Insert ▸ Prominent rather than the dialogue window Format ▸ Paragraph. Most of these tags were already discussed in section 3.5. You may use Insert ▸ Prominent ▸ Indent for adding a fixed amount of indentation to the left margin (this is often needed for computer programs). TeXmacs also provides the jump-in tag (Insert ▸ Prominent ▸ Indent) for typesetting text using a negative first indentation, with the effect of making paragraphs "jump in". A compact layout (Insert ▸ Prominent ▸ Compact) reduces the amount of interparagraph space; see Table 3.5.

The quick brown fox

Jumps over the lazy dog.

The quick fox

Jumps over dogs.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Figure 3.5. Effect of the markup elements indent, jump-in, and compact.

3.10 Page layout

The default page layout is determined by the document style. You can customize it using the dialogue window that is opened via Document ▸ Page. For a newly created document, you may also change some of the settings using the icon ▯ (or possibly ▯, ▯, ▭, ▭, or ▭) on the focus toolbar (see section 2.7). In order to tweak the layout of a specific page, you should rather use Format ▸ Page.

The main characteristics of a page are its size (most of the standard page sizes—such as A4, A5, Letter, Legal, etc.—are supported by TeXmacs) and its orientation (portrait or landscape). These properties can be specified via Document ▸ Page ▸ Format ▸ Page type and Orientation.

Another major issue is how to display pages on your screen. It is crucial to understand that TeXmacs may render your document differently on screen and on paper[3.1]. TeXmacs supports four major rendering modes in Document ▸ Page ▸ Format ▸ Page rendering:

- The paper rendering reflects the actual way your document will be printed. Page breaks are made explicit by showing the pages separately, one below another.

- The papyrus rendering mode treats your document as if it were printed on a big roll of papyrus with the specified "page" width, but whose vertical height is as large as needed.

- The screen setting can be used if you do not intend to print your document. The page width and height are automatically adjusted to the size of your window. This is most useful when browsing web pages or using TeXmacs as an interface for some computer algebra system.

- The beamer mode is reserved for laptop presentations; see chapter 9.

The page breaking algorithm takes some time on large documents, which may hamper the reactivity of the editor while typing. For this reason, we recommend that you edit such documents in "papyrus" mode and only switch to "paper" mode when your document is almost finished (e.g. in order to inspect the page breaks).

3.1. For TeXmacs to be "truly wysiwyg", you need to select the paper rendering mode and Document ▸ Page ▸ Margins ▸ Same screen margins as on paper. However, since screens and paper are different media, editing your document as if it were already printed is not always convenient. For instance, page breaks may heavily change while you are typing, and thereby cause unpleasant jumping around of the cursor. Sentences with page breaks may also become hard to read.

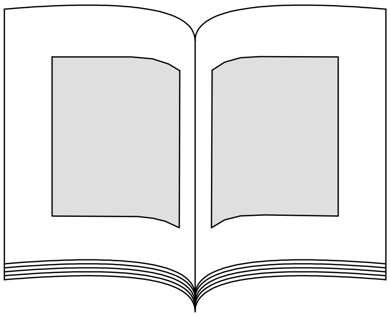


Figure 3.6. Schematic representation of the typical layout of a book: the left margins are larger than the right margins on the (even) left-hand pages, whereas they are smaller on the (odd) right-hand pages.

The second most important decision about page layout concerns the margins; these can be specified in the tab Document ▸ Page ▸ Margins. Remember that the lines of your text should not be made too long, thereby keeping them easy to read. On standard paper of A4 or Letter size, this can be achieved by using sufficiently large margins or a larger font. Another option is to use a two column layout. It is also interesting to know that printed books tend to use different margins for odd and even pages: see Figure 3.6. Since margins on actual paper can be quite large, you finally may wish to use different margins when editing documents on your screen. In Document ▸ Page ▸ Margins, you are therefore allowed to separately specify paper margins and screen margins (or force the same settings for both using Same screen margins as on paper).

The remaining global settings for page layout are intended for more expert users. In particular, it is recommended to *not* specify page headers and footers through Document ▸ Page ▸ Headers, except when you selected the `generic` document style. The reason is that title blocks and section titles often override these settings, since the main title, the authors and titles of sections are typically used inside page headers.

On the other hand, it is safe to customize the headers and footers of a particular page using Format ▸ Page ▸ This page header and This page footer. You may also change the number of a particular page (and all the subsequent ones) using Format ▸ Page ▸ This page number. You may finally change the way page numbers are rendered using Format ▸ Page ▸ Page number rendering.

Chapter 4
Mathematics

You enter *math mode* by starting a new formula or by placing the cursor inside an existing formula. In math mode, the menus, the icon toolbars, and the keyboard behavior are adapted so as to facilitate the insertion of mathematical symbols and markup. For instance, typing - > inserts the arrow →.

Recent versions of TeXmacs also include optional “semantic editing” facilities that will briefly be described at the end of this chapter (see [26, 29] for more details). When used appropriately, these allow you to write documents in which all formulas are at least correct from a syntactical point of view. A “syntax corrector” is included to assist you with this task. Syntactic correctness is for instance important when using formulas as inputs for a computer algebra system. Syntactically correct documents are also less likely to contain “typos”. Further functionalities, such as semantic search and replace, should be developed in the future.

4.1 Incorporating mathematical formulas

TeXmacs provides three main ways to insert mathematical formulas into the main text:

Inline formulas. Short formulas—such as $a^2+b^2=c^2$—are usually embedded directly into the main text flow. Such *inline formulas* can be inserted using $ or Insert ▸ Mathematics ▸ Inline formula.

The typesetter attempts to squeeze inline formulas as much as possible, so that they do not disrupt the general presentation. For example, the presentation $\lim_{n\to\infty} \log n/n=0$ is preferred over $\displaystyle\lim_{n\to\infty} \log n/n=0$, and $x=\frac{1}{2}$ over $x=\dfrac{1}{2}$. Nevertheless, you can force the more voluminous renderings using Format ▸ Display style ▸ on.

Displayed formulas. Big or important formulas are usually displayed on separate lines:

$$x^n+y^n=z^n.$$

Such *displayed formulas* can be inserted using ⌥$ or Insert ▸ Mathematics ▸ Displayed formula and use a less compressed layout than inline formulas. You may turn an inline formula into a displayed one and *vice versa* using ^⇥. Displayed formulas can also be numbered using the toggle ^# or Focus ▸ Numbered.

Equation arrays. For the presentation of multiple equations, it is best to align them as in the following example:

$$x+0 \;=\; x$$

$$\begin{aligned} x+(-x) &= 0\\ x+y &= y+x\\ (x+y)+z &= x+(y+z). \end{aligned}$$

A similar layout is often required for step by step computations

$$\begin{aligned} (\mathrm{e}^{\sin x}+\sin \mathrm{e}^x)' &= (\mathrm{e}^{\sin x})'+(\sin \mathrm{e}^x)'\\ &= (\sin x)'\,\mathrm{e}^{\sin x}+(\mathrm{e}^x)'\sin \mathrm{e}^x\\ &= \mathrm{e}^{\sin x}\cos x+\mathrm{e}^x\sin \mathrm{e}^x, \end{aligned}$$

in which case several left-hand members are simply left empty.

In order to create so-called *equation arrays* of this type, you may use ⌥& or Insert ▸ Mathematics ▸ Several equations. The typesetter uses a table with three columns for the rendering, where the first column is aligned to the right, the third one to the left, and the middle column is centered. New rows are created by pressing ↩. In fact, since equation arrays are special forms of tables, all table editing rules apply (see chapter 5). In a sense, they are also somewhat superfluous: you may obtain the same layout by creating an appropriate table inside a displayed formula. Nevertheless, equation arrays are so common in mathematical documents that it is nice to have special markup for them.

4.2 Mathematical symbols

Following the basic design principles of the user interface (see section 2.1), TEXMACS allows you to enter most mathematical symbols in at least four ways: using the menus, the icon toolbars, appropriate keyboard shortcuts, or LATEX commands.

For example, the binary relation "$\leqslant$" can both be found in the Insert ▸ Symbol ▸ Binary relation menu and in the toolbar menu under the icon $\prec$. When hovering the mouse pointer for a while over the $\leqslant$ in either of these menus, the corresponding keyboard shortcut < = will appear in a help balloon. Users who are familiar with LATEX, may also enter the symbol using \ l e q s l a n t ↩.

The keyboard shortcuts for mathematical symbols were designed according to a small number of simple rules, which make them easy to remember. Before we go into more details, it should also be noticed that symbols carry a precise syntactical semantics in TEXMACS. In particular, TEXMACS carefully distinguishes between so-called *homoglyphs*, which are distinct symbols that look the same. For more information, see sections 4.9 and 4.11 below.

For example, the vertical bar $|$ can be used as a separator for defining sets $R^> = \{x \in R \mid x > 0\}$, but also as the binary relation "divides" (e.g. $11 \mid 1001$), or for restricting the domain of a function: $f|_E$. Such homoglyphs have different binding forces and often come with a different spacing. The most annoying ambiguity is between invisible multiplication xy and function application $\sin x$, which are respectively entered using the shortcuts * and ␣.

As a general rule, we also note that TEXMACS takes care of the spacing inside mathematical formulas. For instance, you enter $a+b$ by typing a + b, not

α	a ⇥	β	b ⇥	γ	g ⇥	δ	d ⇥	ε	e ⇥	ϵ	e ⇥ ⇥ ⇥
ζ	z ⇥	η	h ⇥	θ	j ⇥	ϑ	j ⇥ ⇥ ⇥	ι	i ⇥	κ	k ⇥
$\varkappa$	k ⇥ ⇥	λ	l ⇥	μ	m ⇥	ν	n ⇥	ξ	x ⇥	o	o ⇥
π	p ⇥	ϖ	p ⇥ ⇥ ⇥	ρ	r ⇥	ϱ	r ⇥ ⇥	σ	s ⇥	ς	s ⇥ ⇥
τ	t ⇥	υ	u ⇥	φ	f ⇥	ϕ	f ⇥ ⇥	ψ	y ⇥	χ	q ⇥
ω	w ⇥										
A	A ⇥	B	B ⇥	Γ	G ⇥	Δ	D ⇥	E	E ⇥	Z	Z ⇥
H	H ⇥	Θ	J ⇥	I	I ⇥	K	K ⇥	Λ	L ⇥	M	M ⇥
N	N ⇥	Ξ	X ⇥	O	O ⇥	Π	P ⇥	P	R ⇥	Σ	S ⇥
T	T ⇥	Υ	U ⇥	Φ	F ⇥	Ψ	Y ⇥	X	Q ⇥	Ω	W ⇥

Table 4.1. Keyboard shortcuts for Greek characters.

a ␣ + ␣ b. The space bar is reserved for function application: typing s i n ␣ x produces $\sin x$.

4.2.1 Letter-like symbols

Mathematicians like to use Greek characters as well as letters in several special fonts for particular purposes. For instance, the set of natural numbers is typically denoted by $\mathbb{N}$ and a maximal ideal by $\mathfrak{m}$.

Greek characters can be obtained using the Tab key ⇥, as variants of the usual Roman letters. For instance, you may enter β and Λ using b ⇥ and L ⇥. Table 4.1 shows keyboard shortcuts for the complete Greek alphabet. Notice that the Greek letters ε, θ, κ, π, ρ, σ, and φ admit alternative renderings ϵ, ϑ, $\varkappa$, ϖ, ϱ, ς, and ϕ.

TeXmacs reserves the function keys F5, F6, F7, and F8 for entering characters in special mathematical fonts:

F5. This is the keyboard prefix for upright mathematical symbols. For example, F5 S produces S instead of S, whereas F5 l ⇥ produces λ instead of λ.

F6. This keyboard prefix is used for producing bold letters such as $\boldsymbol{v}$ (F6 v) or $\boldsymbol{S}$ (F6 S). You may combine it with the other prefixes F5, F7, and F8. For example, F6 F5 v yields $\mathbf{v}$ and F6 F7 X yields $\boldsymbol{\mathcal{X}}$.

⇧F6. You may use this prefix for "blackboard bold" fonts. The classical sets $\mathbb{C}$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{Z}$ can for example be obtained using the shortcuts ⇧F6 C, ⇧F6 N, ⇧F6 Q, ⇧F6 R, and ⇧F6 Z. An even easier way to obtain these symbols is using C C, N N, Q Q, R R, and Z Z, i.e. by typing twice the same uppercase letter.

F7. This is the keyboard prefix for calligraphic symbols such as $\mathcal{A}$ (F7 A) and $\mathcal{P}$ (F7 P). Notice that not all fonts provide calligraphic variants $\mathcal{a}$, $\mathcal{b}$, $\mathcal{c}$, … for lowercase letters. Whenever such variants are missing for a given font, then TeXmacs will use a system-dependent substitute font instead (see section 3.8.1).

d	`d ⇥ ⇥`	∂	`d ⇥ ⇥ ⇥`	e	`e ⇥ ⇥`	i	`i ⇥ ⇥`
γ	`g ⇥ ⇥`	λ	`l ⇥ ⇥`	π	`p ⇥ ⇥`	Γ	`G ⇥ ⇥ ⇥`
ℏ	`h ⇥ ⇥`	ı	`i ⇥ ⇥ ⇥`	ȷ	`j ⇥ ⇥`	ℓ	`l ⇥ ⇥ ⇥`
D	`D ⇥ ⇥`	∇	`V ⇥ ⇥`	Δ	`D ⇥ ⇥ ⇥ ⇥`	∞	`@ @`
ℜ	`R E`	ℑ	`I M`	∁	`C ⇥ ⇥ ⇥`	Ⅎ	`G ⇥ ⇥ ⇥ ⇥`
∀	`A ⇥ ⇥`	∃	`E ⇥ ⇥`	∃⃒	`E ⇥ ⇥ ⇥`	℘	`P ⇥ ⇥`
ℵ	`A ⇥ ⇥ ⇥`	ℶ	`B ⇥ ⇥ ⇥`	ℷ	`G ⇥ ⇥`	ℸ	`D ⇥ ⇥ ⇥`
϶	`e ⇥ ⇥ ⇥ ⇥`	ϖ	`w ⇥ ⇥`	Ǝ	`E ⇥ ⇥ ⇥ ⇥`	℧	`W ⇥ ⇥`

Table 4.2. Keyboard shortcuts for several letter-like symbols.

`F8`. This keyboard prefix corresponds to the "Fraktur" font. For example, the shortcuts `F8 m` and `F8 J` produce 𝔪 and ℑ.

In fact, we note that TEXMACS does not consider mathematical letter-like symbols like $\boldsymbol{v}$, ℕ, 𝒜, and 𝔪 as being typeset in a separate font. Instead, such symbols are interpreted as special characters in an extended (UNICODE) alphabet. In particular, the font selector (see section 3.8.2) does not contain any entry for—say—the "blackboard bold" font.

There exist a few more letter-like symbols that cannot be obtained through the systematic mechanisms from above. First of all, the important mathematical constants i, e, and π are entered using `i ⇥ ⇥`, `e ⇥ ⇥`, and `p ⇥ ⇥`. Notice that TEXMACS uses an upright rendering for these constants, which allows you to distinguish them from the letters i, e, and π. The d and λ from differential- and lambda-calculus are entered similarly, using `d ⇥ ⇥` and `l ⇥ ⇥`. Table 4.2 shows the list of keyboard shortcuts for these and other letter-like mathematical symbols.

4.2.2 Other symbols

There are a few simple rules that allow you to enter most non-letter-like mathematical symbols using "natural" key-combinations. The most important rule is juxtaposition: `- >` yields $\rightarrow$, `- - >` yields $\longrightarrow$, and `> =` yields $\geqslant$. Similarly, `| ⇥ -` yields $\vdash$, `| - >` yields $\mapsto$, and `- > < -` yields $\rightleftarrows$. The other rules are all based on the use of a special key:

`⇥`. This is the main key for obtaining *variants* (see section 2.3.2). Some symbols have many variants. For example, `<` yields $<$, `< ⇥` yields $\in$, `< ⇥ ⇥` yields $\subset$, `< ⇥ ⇥ ⇥` yields $\prec$, `< ⇥ ⇥ ⇥ ⇥` yields $\sqsubset$, `< ⇥ ⇥ ⇥ ⇥ ⇥` yields $\langle\rangle$, and `< ⇥ ⇥ ⇥ ⇥ ⇥ ⇥` returns to $<$. You may "cycle back" among the variants using `⇧⇥`, so that `< ⇥ ⇥ ⇧⇥` is equivalent to `< ⇥`. For symbols with many variants, the last variants are obtained most efficiently by cycling back. For example, quick shortcuts for $\langle\rangle$ and $\sqsubset$ are `< ⇧⇥` and `< ⇧⇥ ⇧⇥`.

The variant mechanism is particularly powerful when used in conjunction with juxtaposition. For example, `< =`, `< = ⇥`, `< = ⇥ ⇥`, and `< = ⇥ ⇥ ⇥` respectively yield $\leqslant$, $\leq$, $\leqq$, and $\Leftarrow$. Notice that the juxtapositions are horizontal for the first three shortcuts, but vertical for the last one. Similarly, the shortcuts `< = =`, `< ⇥ ⇥ =`, and `< < < =` produce $\Longleftarrow$, $\subseteq$, and $\lll$.

- **@**. This key is used for putting symbols into circles or boxes. For example, @+ yields $\oplus$ and @x yields $\otimes$. Similarly, @⇥+ yields $\boxplus$.

- **/**. Slashes are used for negations. For example, =/ yields $\neq$ and <=/ yields $\nleq$. Notice that <=⇥⇥/ yields $\nleqq$, whereas <=⇥⇥/⇥ yields $\lneq$.

- **!**. After entering an arrow, this key forces scripts to be placed above or below the arrow. For example, -->^x yields $\longrightarrow^x$, but -->!^x yields $\xrightarrow{x}$.

- **sin**. Special operators such as "sin" are simply obtained by typing sin. More precisely, any sequence of two or more unaccented Latin letters is regarded as an operator by TeXmacs. In order to enter the formula $x y$ (x times y), you have to explicitly insert a multiplication using *. Indeed, typing xy yields xy, whereas x*y produces the desired result. We also recall that TeXmacs makes an explicit distinction between multiplication and function application. This means that you should type L␣x in order to produce $L x$ (L applied to x), and sin␣sin␣x for getting $\sin \sin x$.

Apart from these general rules, there are a few other shortcuts and facts that are worth knowing:

- *****. By default, this key is used for "invisible" multiplications, as in $x y = x \cdot y$. The "visible" multiplication symbols $\times$, $\ast$, $\cdot$, and $\wedge$ (wedge product) are obtained as variants, using *⇥, *⇥⇥, etc.

  Notice that the presence of an invisible multiplication is indicated by a small space. TeXmacs provides a few other invisible operators, such as invisible function application ($\sin x$), invisible addition ($5\,{}^3\!/\!_4$), etc.; see section 4.11. You may use Document ▸ Informative flags ▸ Detailed in order to display additional visual hints that will not appear in print, but that allow you to distinguish between different types of invisible operators while editing.

- **␣**. The space symbol is reserved for function application, as in $\exp x$. By default, TeXmacs tries to avoid the insertion of any spurious space. Typing a␣+␣b is therefore equivalent to typing a+b. If you want to force TeXmacs to insert all spaces you type, then you should select Edit ▸ Preferences ▸ Space bar in math mode ▸ Allow spurious spaces.

  Inserting a space of a specified length is also possible using Format ▸ Whitespace ▸ Horizontal space. From a semantic point of view, TeXmacs ignores all whitespace of this kind. Large spaces that should be considered as function applications or commas can be entered as variants of spaces and commas. For instance, the space in the function application $\varphi \; x$ can be entered using ␣⇥. Notice that this space is larger than the standard function application space in φx.

⊕	@ +	⊖	@ −	⊗	@ *	⊘	@ / ⇥
⊙	@ .	⊚	@ @ ⇥	⊛	@ * ⇥	⦶	@ |
⊞	@ ⇥ +	⊟	@ ⇥ −	⊠	@ ⇥ *	⧄	@ ⇥ /
⊡	@ ⇥ .	⧇	@ ⇥ @	⧆	@ ⇥ * ⇥	◫	@ ⇥ |
±	+ −	∓	− +	×	* ⇥	÷	/ ⇥ ⇥
∗	* ⇥ ⇥	⋆	* ⇥ ⇥ ⇥ ⇥ ⇥	∘	@	●	@ ⇥ ⇥
∩	& ⇥	∪	% ⇥	⊎	% ⇥ +	·	* ⇥ ⇥ ⇥
⊓	& ⇥ ⇥	⊔	% ⇥ ⇥	∨	%	∧	&
⋎	% ⇥ ⇥ ⇥ ⇥	⋏	& ⇥ ⇥ ⇥ ⇥	⊻	% −	⊼	− &
⋉	| ⇥ *	⋊	* ⇥ ⇥ |	⋋	T ⇥ ⇥ ⇥ ⇥	⋌	T ⇥ ⇥ ⇥ ⇥ ⇥

Table 4.3. Binary operators (see also Insert ▸ Symbol ▸ Binary operator).

. The default dot stands for the decimal dot, as in 3.14159. The dot symbol has several variants: the "point" operator from λ-calculus (. ⇥) as in $\lambda x. x^2$, the "dummy" dot · (. ⇥ ⇥) as in $g = f(x, \cdot)$, and the invisible dummy dot (. ⇥ ⇥ ⇥).

Various kinds of ellipses $\ldots$, $\cdots$, $^{\cdots}$, $\vdots$, $\ddots$, and $\iddots$ can be obtained using the shortcuts . ., . . ⇥, etc. The "lower" ellipses should be used in expressions such as $x_1, \ldots, x_n$ and $K[[x_1; \ldots; x_n]]$, whereas the centered ellipses should be preferred in $x_1 + \cdots + x_n$ and $x_1 \cdots x_n$. Notice that TeXmacs provides the additional shortcuts , ,, + +, etc. for entering "$, \ldots ,$", "$+ \cdots +$", etc.

, The comma most commonly serves as a separator, as in $f(x, y, z)$ and $x_1,\ldots,x_n$. Variants are the "decimal comma" (like in 3,14159), the invisible comma (as in the matrix (M_{ij})), and various "wide invisible commas", as in the list of formulas

$$a_1 = b_1 \qquad a_2 = b_1 + b_2 \qquad a_3 = b_1 + b_2 + b_3.$$

&, **%**. The logical connectors ∧ and ∨ are obtained using & and %. The conjunction ∧ has many variants (namely ∩, ⊓, ∐, ⋏, ⅄, ⅄, and &), and so does the conjunction ∨ (namely ∪, ⊔, ∐, ⋎, ⋎, ⋎, ‰, and %). The literal symbols & and % are obtained most efficiently through F5 & and F5 %. Alternative shortcuts for ∧ and ∨ are / \ and \ /, with approximately the same variants as above.

#. This key is used for the "number" prefix #, with variants ♯ and ♮. The "flat" symbol ♭ can be obtained using b ⇥ ⇥.

4.2.3 Keyboard shortcuts for common symbols

For the reader's convenience, we have included Tables 4.3–4.9 with keyboard shortcuts for common symbols. The shortcuts are easily remembered when applying the rules from the previous section. We also note that some of the keyboard shortcuts may change from one version of TeXmacs to another, e.g. due to the appearance of a new symbol.

∼	~	≃	~ -	≈	~ ~	≅	~ =
≁	~ /	≄	~ - /	≉	~ ~ /	≇	~ = /
≍	= ⇥	≡	= ⇥ ⇥	≅	~ =	≋	= ⇥ ⇥ ⇥ ⇥
≭	= ⇥ /	≢	= ⇥ ⇥ /	≇	~ = /	≋̸	= ⇥ ⇥ ⇥ ⇥ /
≏	~ - ⇥ ⇥	≎	~ ~ ⇥	≗	@ =	≖	= @
∽	~ ⇥	⋍	~ - ⇥	∼	F6 ~	≈	F6 ~ ~
≠	= /	∝	@ @ ⇥ ⇥	≊	~ ~ =	≜	@ = ⇥

Table 4.4. Equalities, similarities, and their negations (see also Insert ▸ Symbol ▸ Binary relation).

<	<	⩽	< =	≤	< = ⇥	≦	< = ⇥ ⇥
>	>	⩾	> =	≥	> = ⇥	≧	> = ⇥ ⇥
≪	< <	≪̲	< < =	⋘	< < <	⋘̲	< < < =
≫	> >	≫̲	> > =	⋙	> > >	⋙̲	> > > =
≺	< ⇥ ⇥ ⇥	≼	< ⇥ ⇥ ⇥ =	⪯	< ⇥ ⇥ ⇥ = ⇥	≾	< ⇥ ⇥ ⇥ ~
≻	> ⇥ ⇥ ⇥	≽	> ⇥ ⇥ ⇥ =	⪰	> ⇥ ⇥ ⇥ = ⇥	≿	> ⇥ ⇥ ⇥ ~
⪻	< < ⇥	⪻̲	< < ⇥ =	≺≺≺	< < < ⇥	≺≺≺̲	< < < ⇥ =
⪼	> > ⇥	⪼̲	> > ⇥ =	≻≻≻	> > > ⇥	≻≻≻̲	> > > ⇥ =
⊂	< ⇥ ⇥	⊆	< ⇥ ⇥ =	⫅	< ⇥ ⇥ = ⇥	∈	< ⇥
⊃	> ⇥ ⇥	⊇	> ⇥ ⇥ =	⫆	> ⇥ ⇥ = ⇥	∋	> ⇥
⊏	< ⇥ ⇥ ⇥ ⇥	⊑	< ⇥ ⇥ ⇥ ⇥ =	⋐	< < ⇥ ⇥	⪿	< ⇥ ⇥ +
⊐	> ⇥ ⇥ ⇥ ⇥	⊒	> ⇥ ⇥ ⇥ ⇥ =	⋑	> > ⇥ ⇥	⫀	> ⇥ ⇥ +
⊲	< | ⇥	⊴	< | ⇥ =	⊴	< | ⇥ = ⇥	◀	< | ⇥ ⇥
⊳	| ⇥ >	⊵	| ⇥ > =	⊵	| ⇥ > = ⇥	▶	| ⇥ > ⇥
≲	< ~	⪅	< ~ ~	≾	< ⇥ ⇥ ⇥ ~	⪷	< ⇥ ⇥ ⇥ ~ ~
≳	> ⇥ ~	⪆	> ⇥ ~ ~	≳	> ⇥ ~	⪆	> ⇥ ~ ~

Table 4.5. Comparison relations (see also Insert ▸ Symbol ▸ Binary relation).

∉	< ⇥ /	∌	> ⇥ /				
≮	< /	⩽̸	< = /	≰	< = ⇥ /	⪇	< = ⇥ / ⇥
≯	> /	⩾̸	> = /	≱	> = ⇥ /	⪈	> = ⇥ / ⇥
≨	< = ⇥ ⇥ / ⇥	≨︀	< = ⇥ ⇥ / ⇥ ⇥	⋦	< ~ / ⇥	⪉	< ~ ~ / ⇥
≩	> = ⇥ ⇥ / ⇥	≩︀	> = ⇥ ⇥ / ⇥ ⇥	⋧	> ~ / ⇥	⪊	> ~ ~ / ⇥
⊊	< ⇥ ⇥ = / ⇥	⊊︀	< ⇥ ⇥ = ⇥ / ⇥	⊊	< ⇥ ⇥ = / ⇥ ⇥	⫋	< ⇥ ⇥ = ⇥ / ⇥ ⇥
⊋	> ⇥ ⇥ = / ⇥	⊋︀	> ⇥ ⇥ = ⇥ / ⇥	⊋	> ⇥ ⇥ = / ⇥ ⇥	⫌	> ⇥ ⇥ = ⇥ / ⇥ ⇥
⊀	< ⇥ ⇥ ⇥ /	⋠	< ⇥ ⇥ ⇥ = /	⪯̸	< ⇥ ⇥ ⇥ = ⇥ /	⪵	< ⇥ ⇥ ⇥ = ⇥ ⇥ / ⇥
⊁	> ⇥ ⇥ ⇥ /	⋡	> ⇥ ⇥ ⇥ = /	⪰̸	> ⇥ ⇥ ⇥ = ⇥ /	⪶	> ⇥ ⇥ ⇥ = ⇥ ⇥ / ⇥
⋨	< ⇥ ⇥ ⇥ ~ / ⇥	⪹	< ⇥ ⇥ ⇥ ~ ~ / ⇥	⊏̸	< ⇥ ⇥ ⇥ ⇥ /	⋢	< ⇥ ⇥ ⇥ ⇥ = /
⋩	> ⇥ ⇥ ⇥ ~ / ⇥	⪺	> ⇥ ⇥ ⇥ ~ ~ / ⇥	⊐̸	> ⇥ ⇥ ⇥ ⇥ /	⋣	> ⇥ ⇥ ⇥ ⇥ = /

Table 4.6. Negations of comparison relations (see also Insert ▸ Symbol ▸ Negation).

←	< -	⇐	< = ⇥ ⇥ ⇥	↽	/ - ⇥	↼	/ - ⇥ ⇥
→	- >	⇒	= >	⇁	- / ⇥	⇀	- / ⇥ ⇥
↔	< - >	⇔	< = >	⇋	< - - > ⇥ ⇥ ⇥	⇌	- > < - ⇥
⇇	< - < -	↢	< - <	↩	< - | ⇥	↞	< < -
⇉	- > - >	↣	> - >	↪	| - > ⇥	↠	- > >
⇆	< - - > ⇥ ⇥	⇄	- > < -	↦	| - >		
↫	< - @	↬	@ - >	⇝	~ >	↭	< ~ >
↑	< - ⇥	↕	< - > ⇥	⇑	= > ⇥ ⇥	⇈	< - < - ⇥
↓	< - ⇥ ⇥	⇕	< = > ⇥	⇓	= > ⇥	⇊	< - < - ⇥ ⇥
↿	/ - ⇥ ⇥ ⇥ ⇥	↾	/ - ⇥ ⇥ ⇥	↖	< - ⇥ ⇥ ⇥	↗	- > ⇥ ⇥ ⇥ ⇥
⇃	/ - ⇥ ⇥ ⇥ ⇥ ⇥	⇂	/ - ⇥ ⇥ ⇥ ⇥ ⇥ ⇥	↙	< - ⇥ ⇥ ⇥ ⇥	↘	- > ⇥ ⇥ ⇥

Table 4.7. Arrows (see also Insert ▸ Symbol ▸ Arrow).

⟵	< - -	⟸	< = =	⟻	< - - |	$\hookleftarrow$	< - - | ⇥
⟶	- - >	⟹	= = >	⟼	| - - >	$\hookrightarrow$	| - - > ⇥
⟷	< - - >	⟺	< = = >	—	- -	=	= =
↑	< - - ⇥	↓	< - - ⇥ ⇥	⇑	< = = ⇥	⇓	< = = ⇥ ⇥
↥	< - - | ⇥ ⇥	↧	< - - | ⇥ ⇥ ⇥ ⇥	⤴	< - - | ⇥ ⇥ ⇥	⤵	< - - | ⇥ ⇥ ⇥ ⇥ ⇥

Table 4.8. Long arrows (see also Insert ▸ Symbol ▸ Arrow).

¬	! ⇥	∅	@ /	△	@ ⇥ ⇥ ⇥ ⇥	▲	@ ⇥ ⇥ ⇥ ⇥ ⇥
∘	@	□	@ ⇥	●	@ ⇥ ⇥	■	@ ⇥ ⇥ ⇥
⊤	T ⇥ ⇥	⊥	T ⇥ ⇥ ⇥	⊢	| ⇥ -	⊣	- | ⇥
⊩	| | ⇥ -	⊪	| | | ⇥ -	⊨	| ⇥ =		
♭	b ⇥ ⇥	♯	# ⇥	♮	# ⇥ ⇥		
♦	< > ⇥	♥	< > ⇥ ⇥	♠	< > ⇥ ⇥ ⇥	♣	< > ⇥ ⇥ ⇥ ⇥
…	. .	⋯	. . ⇥	˙˙˙	. . ⇥ ⇥		
⋮	. . ⇥ ⇥ ⇥	⋱	. . ⇥ ⇥ ⇥ ⇥	⋰	. . ⇥ ⇥ ⇥ ⇥ ⇥		

Table 4.9. Miscellaneous symbols (see also Insert ▸ Symbol ▸ Miscellaneous).

4.2.4 When a symbol is hard to find

For certain symbols like ✠, it may not be clear how to obtain it using the systematic mechanisms described in this section. Since the symbol vaguely reminds one of addition, you may take your chances and try all variants of +. If you are unlucky, or if the symbol is so exotic (e.g. ♫) that it reminds you of nothing at all, then you have several fall-back options:

TeXmacs menus. You may try to look it up in the submenus of Insert ▸ Symbol and especially in Insert ▸ Symbol ▸ Miscellaneous.

System tools. Most operating systems integrate tools for visualizing UNICODE characters and copy and paste them to other applications. For example, MACOS integrates "Character Viewer" for this purpose.

LATEX names. If you know the LATEX name of the symbol, then you may enter it using that name. For example, ♫ corresponds to the LATEX command `\twonotes`, so you may obtain this symbol via `\ t w o n o t e s ↩`.

Internal TEXMACS names. Most mathematical symbols are internally known under the same name as under LATEX. Symbols can directly be entered using their TEXMACS name via the keyboard shortcut `^q`. Hence `^q t w o n o t e s ↩` also produces ♫.

UNICODE numbers. You may also look up the UNICODE number of your symbol inside UNICODE tables that you can find on the web. The hexadecimal UNICODE number prefixed by # can also be used as an internal TEXMACS name for the symbol. For example, the symbol ♫ corresponds to the UNICODE character 266B, so you may enter it by typing `^q # 2 6 6 B ↩`.

4.3 Two-dimensional mathematical markup

Subscripts and superscripts. Subscripts and superscripts are created using the special keys `_` and `^`. For example, you can produce a_n by typing `a _ n` and a^{a^x} via `a ^ a ^ x`. Notice that you have to explicitly move the cursor out of the script whenever needed, usually by pushing `→` once. In other words, $a_n + b_n$ is obtained through `a _ n → + b _ n`, whereas `a _ n + p` yields a_{n+p}.

Subformulas may simultaneously carry a subscript and a superscript. For instance, a_n^2 is obtained using `a _ n → ^ 2`. The order in which a subscript and a superscript are entered does not affect the rendering, but is important for the semantics: if we entered the above formula a_n^2 using `a ^ 2 → _ n`, then it would mean $(a^2)_n$ instead of $(a_n)^2$.

TEXMACS also allows you to put subscripts and superscripts at the left-hand side of an expression, using the shortcuts `⌥l _` and `⌥l ^`. For example, you may enter the formula ${}^{\pi}x$ by typing `⌥l ^ p ⇥ → x`. A formula may be surrounded by as many as four subscripts and superscripts, as in ${}^{*}_{*}\mathrm{He}^{*}_{*}$.

In some cases, subscripts and superscripts are put below or above a given expression, rather than to its right. This is typically so for big and limit-like operators in displayed equations:

$$\lim_{n\to\infty} \sum_{k=0}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}.$$

It is also possible to explicitly typeset a script above or below another expression using the shortcuts `⌥a` and `⌥b`:

$$f^{\circ n} = f \circ \overset{n\times}{\cdots} \circ f \qquad\qquad x = \underset{n\to\infty}{\mathrm{statlim}}\, x_n.$$

Shortcut	Purpose	Inline	Displayed
⌥f	Ordinary fractions	$\tfrac{a}{b+c}$	$\dfrac{a}{b+c}$
⌥f ⇥	Inline fractions	$\tfrac{a}{b+c}$	$\tfrac{a}{b+c}$
⌥f ⇥ ⇥	Displayed fractions	$\dfrac{a}{b+c}$	$\dfrac{a}{b+c}$
⌥f ⇥ ⇥ ⇥	Slashed fractions	$^1/_2$	$^1/_2$
⌥f ⇥ ⇥ ⇥ ⇥	Continued fractions	$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\ddots}}}$	$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\ddots}}}$

Table 4.10. Various blends of fractions.

Primes. Single and double primes can be entered using the keys ' and ". Primes behave much in the same way as superscripts, but it is slightly more efficient to insert or remove them (compare entering $a'+b$ and a^2+b using a ' + b *versus* a ^ 2 → + b). Using `, you may also insert a "left backprime", as in $`a$.

There are various alternative types of primes, which can be obtained as variants: the primes in $a^{\backprime}$, a^*, $a^\dagger$, $a^\ddagger$, and $a^{✱}$ are entered using the shortcuts ' ⇥, ' ⇥ ⇥, etc. You may for example write V^* for the dual of V and $f^\dagger$ for the logarithmic derivative f'/f of a function f. Backprimes and double primes also have several variants. For instance, you may enter V^{**} and $^{✱}a$ using V " ⇥ ⇥ and ` ⇧⇥ a.

Fractions. TeXmacs uses ⌥ as the principal modifier key for the insertion of mathematical markup. You may for example insert a fraction using ⌥f (or Insert ▸ Fraction). There are several kinds of fractions, which are listed in Table 4.10. Alternatively, fractions may use a one-dimensional layout as in a/b, $15:5$, and $15 \div 3$. The symbols /, :, and ÷ are obtained through /, / ⇥, and / ⇥ ⇥.

Square roots and n-th roots. You may enter a square root using ⌥s (or Insert ▸ Square root) and an n-th root using ⌥s ⇥ (or Insert ▸ N-th root). Hence, ⌥s x yields $\sqrt{x}$ and ⌥s ⇥ ← ← 3 yields $\sqrt[3]{x}$.

The degree n of an n-th root can also be added *a posteriori* using ⌥→. For example, assume that your cursor is at the indicated position in $\sqrt{x|}$. Then typing ⌥→ leads you to the state $\sqrt[|]{x}$. Conversely, pressing ⌫ in the latter situation will remove the degree and bring you back to the original state.

Notice that the sqrt tag for square roots and n-th roots is an example of a tag with a variable number of arguments (either one or two in this case). The above keyboard shortcuts ⌥→ and ⌫ can be used more generally in such situations for the insertion and removal of arguments: see section 10.5.

Ordinary text. Ordinary text can be inserted into a formula using `⌥$` or Insert ▸ Text. This is what we did for the text " is sufficiently large" below:

$$L = \{x | x \text{ is sufficiently large}\}.$$

Negations. Some of the TeXmacs relation symbols do not admit any negated versions. You may manually negate a symbol or a formula using `⌥n` or Insert ▸ Negation:

$$a \not{\mathcal{R}} b \iff \neg(a \mathcal{R} b).$$

4.4 Big operators

Big operators—such as integration $\int$ and n-ary summation $\sum$—are entered using the keyboard prefix `⇧F5` or from the Insert ▸ Symbol ▸ Big operator menu. Table 4.11 lists the available big operators together with the corresponding keyboard shortcuts. The rendering of big operators is quite different for inline and displayed formulas: compare $\sum_{k=1}^{\infty} 1/k^2$ and $\int_0^1 f(x)\,\mathrm{d}x$ with

$$\sum_{k=1}^{\infty} \frac{1}{k^2} \qquad \int_0^1 f(x)\,\mathrm{d}x.$$

The big integral signs has two variants, depending on where you want to place subscripts and superscripts in display mode. By default, the scripts are placed to the right:

$$\int_0^{\infty} \frac{\mathrm{d}x}{1+x^2}.$$

The alternative rendering "with limits"

$$\int\limits_0^{\infty} \frac{\mathrm{d}x}{1+x^2}.$$

can be obtained using the `!` postfix, by typing `⇧F5` `I` `!`. Some people also prefer integrals to be rendered using an upright font. This gives rise to two further variants. For instance, you may obtain the triple integral in

$$\iiint\limits_V f(x,y,z)\,\mathrm{d}x \wedge \mathrm{d}y \wedge \mathrm{d}z$$

using the shortcut `⇧F5` `I` `I` `I` `⇥` `⇥` `⇥`. We recall that it is good practice to enter the d's of the differentials using `d` `⇥` `⇥`, so as to distinguish them from the letter d in $a+b+c+d$. Similarly, the exterior product was entered using `*` `&`.

Shortcut	Result	Shortcut	Result	Shortcut	Result
⇧F5 I	$\int$	⇧F5 I I	$\iint$	⇧F5 I I I	$\iiint$
⇧F5 O	$\oint$	⇧F5 O O	$\oiint$		
⇧F5 A	$\coprod$	⇧F5 P	$\prod$	⇧F5 S	$\sum$
⇧F5 @ +	$\bigoplus$	⇧F5 @ *	$\bigotimes$	⇧F5 @ .	$\bigodot$
⇧F5 U	$\bigcup$	⇧F5 N	$\bigcap$	⇧F5 U +	$\biguplus$
⇧F5 U ⇥	$\bigsqcup$	⇧F5 N ⇥	$\bigsqcap$	⇧F5 U ⇥ ⇥	$\Box$
⇧F5 V	$\bigvee$	⇧F5 V ⇥	$\bigcurlyvee$	⇧F5 V ⇥ ⇥	$\bigtriangledown$
⇧F5 W	$\bigwedge$	⇧F5 W ⇥	$\bigcurlywedge$	⇧F5 W ⇥ ⇥	$\bigtriangleup$
⇧F5 B	$\|\|$	⇧F5 B ⇥	$\|\|\|$		

Table 4.11. Big mathematical operators.

4.5 Large delimiters

Brackets inside mathematical formulas should always match. As soon as you open a bracket "(", TEXMACS therefore automatically inserts the corresponding closing bracket ")". Occasionally, it may occur that you do not want the closing bracket, or that you want to replace it by another kind of bracket. This can be done as follows:

- If your cursor is just before the closing bracket inside (a,b), then pressing] will turn the expression into $(a,b]$. More generally, this mechanism can be used to turn any closing bracket into one of any other form.
- If your cursor is just behind the closing bracket inside $\{{a \atop b}\}$, then pressing ⌫ will remove the closing bracket, yielding $\{{a \atop b}$. Similarly, in order to remove an opening bracket, put the cursor just in front of it and press ⌦. Notice that pressing ⌫ or ⌦ in the example $f()$ will remove both brackets.

Notice that the automatic insertion of matching closing brackets can be disabled using Edit ▸ Preferences ▸ Keyboard ▸ Automatic brackets ▸ Disabled. It should also be noticed that the closing curly bracket was not really removed in the above formula $\{{a \atop b}$: TEXMACS rather replaced it with an "invisible" closing bracket (see section 4.11), thereby ensuring that all brackets remain matching. Selection of the preference Automatic brackets ▸ Disabled has a similar effect of replacing visible closing brackets by invisible ones.

For some delimiters, such as $|$, the opening and closing versions coincide. For instance, entering a vertical bar | will produce an absolute value. The vertical bar is often used as a separator as well. You may insert a small bar-separator $|$

using `|` `⇥` or `F5` `|`. Large bar-separators are entered using `⌥m` `|` or `|` `⇥` `⇥`; they are used for producing the vertical bars in formulas like

$$\left\langle \frac{a}{b+c} \middle| \frac{p}{q+r} \middle| \frac{a}{b+c} \right\rangle.$$

We notice that such large separators do not exist in (LA)TEX. There may be as many middle delimiters between a left and a right delimiter as needed and they are all scaled to the same appropriate size. Notice that not all vertical bars are delimiters. For example, the binary relation "divides" is entered using `|` `⇥` `⇥` `⇥` `⇥` or `F5` `|` `⇥` `⇥`; we will see more examples in section 4.11.

In TEXMACS, large delimiters may either be "left delimiters", "right delimiters" or "middle delimiters". By default, (, [, {, and ⟨ are left delimiters, whereas),], }, and ⟩ are right delimiters. But their status can be changed using the `⌥l`, `⌥r`, and `⌥m` prefixes. For example, `⌥l` `)` produces), considered as a large left delimiter, together with the corresponding "closing bracket" (. Open French-style intervals $]a,b[$ can therefore be entered using `⌥l` `]`.

By default, the sizes of the brackets are adjusted to the expression between the brackets. Small delimiters, created using the `⌥`-prefix, keep their sizes independently of the enclosed expression. Thus, the brackets in the left-hand and right-hand expressions below were entered respectively using `(` and `⌥(`:

$$f\left(\sqrt{e^{e^{e^x}}+1}\right) \qquad f(\sqrt{e^{e^{e^x}}+1}).$$

You may also use `^*` in order to toggle between large and small delimiters.

Sometimes you may want large delimiters of a particular size, instead of the self-adjusting ones. This can be achieved in two main ways. The best strategy is usually to tweak the size of the expression between the brackets using the Format ▸ Adjust menu. For example, if you think that the brackets in the formula

$$f\left(\frac{x}{y}\right)$$

are slightly too large, then you may reduce the size of the fraction using Format ▸ Adjust ▸ Reduce, which yields

$$f\left(\tfrac{x}{y}\right).$$

TEXMACS also allows you to directly increase or decrease the size of the brackets themselves using the keyboard shortcuts `⌘↑` and `⌘↓`. In order to reset to the initial size, you may use `⌘⌫`.

We finally notice that it is possible to insert a pair of invisible brackets using `⌥l` `.`. This is for instance useful in computational contexts, when formulas should have a precise, not merely visual semantics. Alternatively, you may use `⌘=` to confine a selected formula inside a "horizontal group". This additionally prevents the formula from being hyphenated.

Shortcut	Example	Wide example	Shortcut	Example	Wide example
⌥~	$\tilde{x}$	$\widetilde{x+y}$	⌥B	$\bar{x}$	$\overline{x+y}$
⌥^	$\hat{x}$	$\widehat{x+y}$	⌥C	$\check{x}$	$\widecheck{x+y}$
⌥A	$\overset{\frown}{x}$	$\overparen{x+y}$	⌥U	$\breve{x}$	$\overset{\smile}{x+y}$
⌥V	$\vec{x}$	$\overrightarrow{AB}$	⌥o < >	$\overleftrightarrow{x}$	$\overleftrightarrow{ABCD}$
⌥o <	$\overleftarrow{x}$	$\overleftarrow{ABCD}$	⌥o >	$\overrightarrow{x}$	$\overrightarrow{ABCD}$

Table 4.12. Keyboard shortcuts for wide mathematical accents.

4.6 Wide accents and extensible arrows

Mathematical accents are created using the modifier key ⌥ or the prefix ⌥o (as in Over). For example, $\hat{\varphi}$ is entered by typing ⌥^ f ⇥, and $\overrightarrow{ABCD}$ using ⌥o > A B C D. Many of these accents stretch with the formulas below them, as in $\widetilde{\varphi+\psi}$. The list of all such *wide accents* is given in Table 4.12. Other accents never stretch with the formulas below them: see Table 4.13.

The same accents may be inserted below the expressions using the ⌥u prefix (Under). Hence $\underline{x+y}$ is entered using ⌥u B x + y and $\underleftrightarrow{ABCD}$ is obtained by typing ⌥u < > A B C D.

Shortcut	Example	Shortcut	Result
⌥.	$\dot{x}$	⌥'	$\acute{x}$
⌥"	$\ddot{x}$	⌥`	$\grave{x}$
⌥" ⇥	$\dddot{x}$	⌥@	$\mathring{x}$
⌥" ⇥ ⇥	$\ddddot{x}$		

Table 4.13. Keyboard shortcuts for non-stretchable accents.

So-called *overbraces* and *underbraces* are entered in a similar way as wide accents. They have the additional property that superscripts and subscripts are placed above and below them. In Table 4.14, we have listed the available overbraces. The expression below can therefore be obtained by typing ⌥o { a + b + c → ^ S ⇥:

$$\overbrace{a+b+c}^{\Sigma}.$$

Shortcut	Example	Shortcut	Example
⌥o {	$\overbrace{x+y+z}$	⌥o }	$\overset{\underbrace{\qquad\quad}}{x+y+z}$
⌥o (	$\overparen{x+y+z}$	⌥o)	$\overset{\underparen{\qquad\quad}}{x+y+z}$
⌥o [	$\overbracket{x+y+z}$	⌥o]	$\overset{\underbracket{\qquad\quad}}{x+y+z}$

Table 4.14. Wide overbraces.

$\overset{a+b+c}{\rule{3em}{0.4pt}}$	$\xrightarrow{a+b+c}$	$\xleftarrow{a+b+c}$	$\xleftrightarrow{a+b+c}$	$\xmapsto{a+b+c}$
$\xlongequal{a+b+c}$	$\xRightarrow{a+b+c}$	$\xLeftarrow{a+b+c}$	$\xLeftrightarrow{a+b+c}$	

Table 4.15. Available extensible arrows.

Besides delimiters and accents, some of the arrows are also extensible in TeXmacs. This is typically needed whenever they come with a superscript above them (or a subscript below them). For instance, the following formula was entered by typing `E - > ^ f @ * g → F`:

$$E\xrightarrow{f\otimes g}F.$$

Here `- > ^` was used as a keyboard shortcut for creating an extensible arrow with a superscript above it. Whenever you need a non-extensible arrow with an ordinary superscript (in the upper-right corner), the trick is to "break" the keyboard shortcut; e.g. `E - > ← → ^ + → F` yields

$$E\rightarrow^{+}F.$$

Table 4.15 lists the available extensible arrows in TeXmacs.

4.7 Matrices and mathematical tables

New 1×1 matrices can be entered using Insert ▸ Table ▸ Matrix, `⌘t N m` (Table New Matrix), `( ⌘t N T` (Table New Table), or `\ m a t r i x ↩`. Matrices are special kinds of tables, which means that the full range of table editing facilities is available (see chapter 5). For now, it suffices to know that new rows and columns can be inserted using the shortcuts `↩`, `⌥→`, `⌥←`, `⌥↑`, and `⌥↓`. For instance, the matrix

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

can be entered using `⌘t N m a ⌥→ b ↩ c → d`. Empty rows and columns can be removed simply by pressing `⌫`.

There are a few other kinds of table-like structures that are useful in mathematics, starting with the following variants of matrices: determinants (`⌘t N d` or `| ⌘t N T`), matrices with angular brackets (`⌘t N M` or `[ ⌘t N T`), and ordinary centered tables (`⌘t N T`), which correspond to matrices with no brackets at all:

$$\begin{pmatrix} a & b\\ c & d \end{pmatrix} \qquad \begin{vmatrix} a & b\\ c & d \end{vmatrix} \qquad \begin{bmatrix} a & b\\ c & d \end{bmatrix} \qquad \begin{matrix} a & b\\ c & d \end{matrix}.$$

Note that you may cycle among the first three variants using `^⇥`. Secondly, you may create so-called "choice lists" using `⌘t N c` or `{ ⌘t N T`, as in

$$f(x)=\begin{cases} 0 & \text{if } x\leqslant 0\\ \mathrm{e}^{-\frac{1}{x^2}} & \text{otherwise.} \end{cases}$$

Finally, TeXmacs provides the stack tag for tables in which the spacing is reduced to the strict minimum (⌘t N s). This is useful when tables are used in subscripts or superscripts:

$$S = \sum_{\substack{1\leqslant i\leqslant n\\1\leqslant j\leqslant m}} \varphi(x_i, y_j).$$

4.8 Commutative diagrams

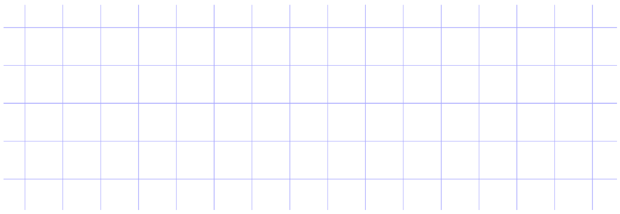
The most appropriate way to create commutative diagrams depends on the required level of complexity. The simplest kind of diagrams can be constructed as centered tables:

$$\begin{tikzcd}
A\amalg B & A \arrow[l, "\iota_A"'] \arrow[d, "\alpha"] \\
B \arrow[u, "\iota_B"] \arrow[r, "\beta"'] & X
\end{tikzcd} \tag{4.1}$$

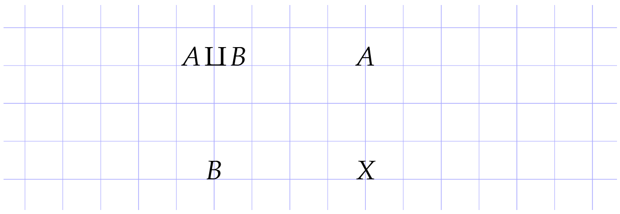
Here we recommend the use of long horizontal and vertical arrows. The small scripts ι_B and α beside the long vertical arrows were created using the markup elements left-script and right-script.

Centered tables may quickly become problematic when you need diagonal or curved arrows. The most general approach to commutative diagrams is to consider them as pictures. For general information about how to draw technical pictures, we refer to chapter 8. In this section, we will describe those features that are useful for creating the same commutative diagram (4.1) as above.

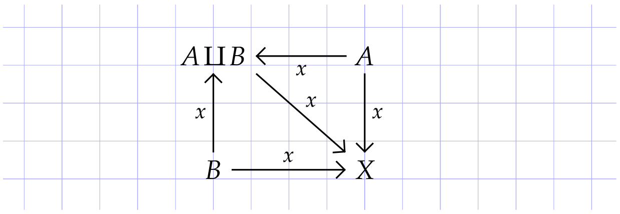
Inside a mathematical formula, we start with the creation of an empty commutative diagram using Insert ▸ Image ▸ Commutative diagram. This inserts a small grid of the following type:



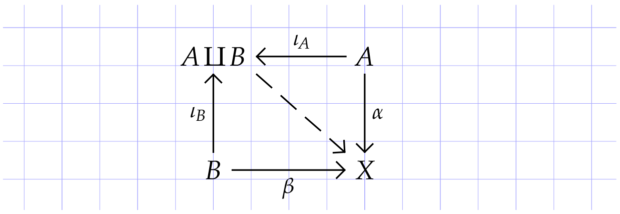
In order to ensure a uniform design, the main nodes $A\amalg B$, A, B, and X should be placed at points on the grid, using the default alignment (namely: horizontally centered and vertically aligned at the baseline):



At the next stage, we want to add the arrows. $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ provides a special mode for adding arrows surrounded by text: Insert ▸ Arrow with text. Furthermore, when moving the mouse around the four formulas, you will observe that there are eight special positions around each formula that "attract" the pointer. These special points should be used as the starting and end points of your arrows:



For the diagonal arrow, a dashed line style would actually have been more appropriate. Such alternative line styles can be selected using Focus ▸ Line dashes. At the next stage, you should select Insert ▸ Mathematics and edit the formulas next to the arrows. You may use ^⇥ to put the text at the other side of an arrow.



In order to finalize the commutative diagram, it now suffices to remove the grid and "crop" the diagram using Insert ▸ Grid ▸ Default and Insert ▸ Geometry ▸ Crop ▸ Crop:

$$\begin{tikzcd} A \amalg B \arrow[rd, dashed] & A \arrow[l, "\iota_A"'] \arrow[d, "\alpha"] \\ B \arrow[u, "\iota_B"] \arrow[r, "\beta"'] & X \end{tikzcd} . \qquad (4.2)$$

If the diagram needs further editing, then you may restore the grid using Insert ▸ Grid ▸ Notebook.

A few further remarks are in order. First of all, $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ does *not* automatically adjust the endpoints of the arrows if you edit one of the formulas A, B, X, or $A \amalg B$. On the other hand, the position of the formula next to the arrow is automatically updated when modifying the endpoints. In documents with several commutative diagrams, we also recommend to consistently opt for either one of the two styles exemplified in (4.1) and (4.2).

4.9 Semantics of mathematical formulas

Recent versions of $\text{T}_\text{E}\text{X}_{\text{MACS}}$ can help you to write documents in which all formulas are at least correct from a syntactical point of view. For example, in the formula $a+b$, the computer will understand that + is an operator that applies to the arguments a and b. It will also flag $a+$ as being incorrect. In fact, the "semantics" that we aim at does not go any further: $\text{T}_\text{E}\text{X}_{\text{MACS}}$ is unaware of the mathematical nature of addition.

Semantic editing does require additional efforts from the user, or at least a little adaptation. For instance, it is the user's job to enter multiplications using the shortcut `*` and function applications using `␣`. These operations cannot be distinguished from their appearance, since they are both printed as invisible whitespace. However, the semantics of these operations is clearly very different.

Although semantically correct documents are usually not very different from informal presentation-oriented documents as far as typesetting is concerned, the additional user effort may pay off for several reasons:

- Adequate semantics is a prerequisite when using formulas as input for a computer algebra system.
- Syntactically correct documents are less likely to contain "typos" or more intricate mathematical errors.
- For certain editing operations, such as cut and paste, one may directly select subformulas that are meaningful from the syntactical point of view.
- It reduces the risk of using non-standard notations that will be difficult to understand for potential readers of your work.

Furthermore, other semantic facilities might be integrated in the feature, such as semantic search and replace, or semantic search on the web.

In order to activate the semantic editing facilities, you must toggle Edit ▸ Preferences ▸ Mathematics ▸ Semantic editing. In the semantic editing mode, several of the structured editing features of $\text{T}_\text{E}\text{X}_{\text{MACS}}$ apply to the syntactic structure of the formula, rather than the visual structure of the document. For instance, the *semantic focus* is usually a subformula of the current focus. Similarly, only syntactically meaningful subformulas can be selected when making a selection.

The semantic focus is useful for several reasons. First of all, it is displayed in green if the formula is syntactically correct and in red if you made an error. This allows to quickly notice any typos while entering a formula. Secondly, if you have any doubt on the precedence of a mathematical operator or relation, then the semantic focus tells you the default interpretation: by putting your cursor right next to your operator, the subexpression to which the operator applies will be highlighted. In the case of an addition, or a more general associative operator, all summands are highlighted.

4.10 Common errors and syntax corrections

By default, the semantic editing mode "understands" most classical mathematical notations. This is achieved through the use of a carefully designed grammar for mainstream mathematics. Obviously, the use of a fixed grammar may cause the following problems:

- Mathematical formulas frequently contain *ad hoc* notations. For instance, the formulas might contain some text or meaningful whitespace. Another example of an *ad hoc* notation is the sign sequence $++-+-+$. In such cases, the user should explicitly annotate the appropriate parts of the formula in order to make them semantically meaningful; see section 4.12.
- The TeXmacs grammar used for the interpretation of mathematical formulas may be incomplete or inadequate for certain situations. It is possible to customize or extend the grammar using the standard TeXmacs macro mechanism (see section 4.12). Notations for specific areas may be grouped together in dedicated style packages.

Besides these intrinsically hard-to-avoid problems, the following common mistakes are a further source of trouble for associating semantics to mathematical formulas:

- Since TeXmacs is a wysiwyg editor, some of the structure of the document is invisible to the user. For example, the presence of a mathematical formula $x+y$ is indicated through the use of an italic slant and special spacing. However, in the formula $f(x)$ it is easy to type the closing bracket outside the formula, with no visual difference.
- Various mathematical notations are visually ambiguous. For example, $a\,(b+c)$ would usually be understood as $a\cdot(b+c)$, whereas $f(x+y)$ typically corresponds to a function application. In the semantic editing mode, the user is expected to resolve this ambiguity by hand by entering multiplications using `*` and spaces using `␣`. The multiply/apply ambiguity is one of the main sources of syntax errors, since many users do not pay attention to invisible differences. Similarly, the $\wedge$ glyph could be the "logical and" or the "wedge product". This "homoglyph" issue will be addressed in more detail in section 4.11 below.
- It could be that a text was originally written in LaTeX or an old version of TeXmacs. In that case, the document contains no special indication on matching brackets or the scopes of big operators. For example, in the formula $[x,y[$, should we interpret the second bracket as a closing bracket? This is indeed the standard French notation for an interval with an open right end. More generally, all problems that we have mentioned so far tend to be present simultaneously when trying to associate semantics to existing documents.

After activation of the semantic editing mode, you may check whether a formula is correct by positioning your cursor inside it and looking at the color of the bounding box of the semantic focus: a green color corresponds to a correct formula and a red color indicates an error in the formula. Alternatively, assuming that the focus is on a mathematical formula, you may select Focus ▸ Preferences ▸ Highlight incorrect formulas, in which all incorrect formulas are highlighted inside red boxes.

For the second kind of common errors, TeXmacs includes an automatic syntax corrector. Assuming that your cursor is inside a formula, you may use Edit ▸ Correct ▸ Correct all for the correction of all formulas in your document, or the correction of the current selection. If the versioning tool is activated (see section 10.9), then you may use Edit ▸ Correct ▸ Correct manually to show the differences between the original and the corrected versions. You may then use the versioning tool to go through these differences and select the preferred versions.

The precise algorithms that are used for the correction may be enabled or disabled from Edit ▸ Preferences ▸ Mathematics ▸ Manual correction:

Remove superfluous invisible operators. This algorithm is used in order to remove any superfluous function applications or multiplications. For instance, users who are accustomed to editing ASCII files often type spaces around binary infixes such as addition. Such "function applications" will be removed by this algorithm.

Insert missing invisible operators. In LaTeX, multiplications and function applications are never entered explicitly. When importing a LaTeX document, it is therefore important to detect and insert missing multiplications and function applications.

Homoglyph substitutions. This algorithm may perform some other useful substitutions of symbols by visually similar, but semantically distinct symbols. For instance, whenever appropriate, the backslash symbol \ is replaced by the binary "set difference" infix (as in $X \setminus Y$).

From the Edit ▸ Preferences ▸ Mathematics ▸ Automatic correction, you may also select the correction algorithms that should be applied automatically whenever you open a file. These corrections are always carried out when importing a LaTeX file.

After syntax correction, the remaining errors indicate genuine typos at worst, or non-standard or unsupported notations at best. We also note that "correct" formulas do not necessarily have the intended meaning. In order to check whether the operators indeed apply to the intended arguments, you should keep an eye on the current focus while typing your formulas.

4.11 Semantics of mathematical symbols

The mathematical symbols in TEXMACS all come with a certain number of properties that correspond to their intended meaning. For example, TEXMACS is aware that "+" is an infix operator, whereas "!" is rather a postfix, and "," a separator.

TEXMACS has special symbols $\mathrm{e} = 2.71828\cdots$, $\pi = 3.14159\cdots$, and $\mathrm{i} = \sqrt{-1}$ for important mathematical constants. These constants are displayed "upright" and they can be entered using the shortcuts e ⇥ ⇥, p ⇥ ⇥, and i ⇥ ⇥. Note that ordinary variables using the same letters are rendered in italics, as e, π, and i.

However, semantically distinct symbols may display in a similar way. For instance, the comma separator, as in $f(x,y)$, is different from the decimal comma, as in $3{,}14159\cdots$. Notice that the two symbols admit different spacing rules. Semantically distinct symbols that are rendered using the same glyph are called *homoglyphs*. Notice that our semantics is purely syntactic: for instance, the + infix is commonly used for addition, but sometimes also for the concatenation of strings. Nevertheless, these two uses do not differ from a syntactical point of view, since the + symbol remains a binary infix operator with the same precedence with respect to other operators.

The most confusing homoglyphs are the various invisible symbols supported by TEXMACS:

- The multiplication, entered by *. Example: $a\,b$.
- Function application, entered by ␣. Example: $\sin x$.
- An invisible separator, entered by , ⇥ ⇥. Example: $A = (a_{ij})$.
- An invisible addition, entered by + ⇥ ⇥ ⇥ ⇥. Example: $17\,{}^3\!/_8$.
- An invisible symbol, entered by . ⇥ ⇥ ⇥. Examples: the increment operator $\;+1$, the zeros in a diagonal matrix

$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix},$$

  or the omitted exponent one in $x_1^3 - x_1^2 + x_1 - 1$ (using an invisible exponent for the degree one term allowed us to vertically align all subscripts).
- An invisible bracket (mainly for internal use). A matching pair of invisible brackets is entered using (⇥.

We recommend in particular that you make it a habit to systematically distinguishing between multiplication and function application: we already mentioned the ambiguity $a(b+c)$ *versus* $f(x+y)$), which cannot be resolved automatically.

Shortcut	Glyph	Example	Semantics
*		$a\,b$	Multiplication
␣		$\sin x$	Function application
, ⇥ ⇥		$a_{ij} = a_{ji}$	Invisible separator
+ ⇥ ⇥ ⇥ ⇥		$17\,{}^3/_8$	Invisible addition
. ⇥ ⇥ ⇥		$+1 : x \mapsto x+1$	Invisible symbol
(⇥		$\Phi \equiv \forall x, P(x)$	Invisible bracket(s)
|	|	$\vert -x \vert = \vert x \vert$	Absolute value
| ⇥	|	$\langle a \vert b \vert c \vert d \rangle$	Separating bar
| ⇥ ⇥	|	$\left\langle \frac{v+w}{2} \middle\vert \frac{u-\bar{u}}{2} \right\rangle$	Extensible middle bar
| ⇥ ⇥ ⇥ ⇥	|	$\{x \in \mathbb{R} \mid x > 0\}$	Such that bar
| ⇥ ⇥ ⇥ ⇥ ⇥	|	$11 \mid 1001$	Divides relation
,	,	$f(x,y)$	Comma separator
, ⇥	,	123,456	Decimal comma
.	.	123.456	Decimal point
. ⇥	.	$\lambda x. x^2$	Dot connector
* ⇥ ⇥ ⇥	·	$\boldsymbol{v} \cdot \boldsymbol{w}$	Dot multiplication
. ⇥ ⇥	·	$\cdot + 1$	Dummy wildcard
:	:	$a{:}b{:}c$	Separator
: ⇥	:	$x : \mathsf{Int}$	Type satisfaction
/ ⇥	:	$121 : 11 = 11$	Division
: ⇥ ⇥ ⇥	:	$\{x \in E : P(x)\}$	Such that colon
\ ⇥	\	$\backslash x$	Backslash
\ ⇥ ⇥	\	$\mathbb{N}^{>} = \mathbb{N} \setminus \{0\}$	Set minus
&	∧	$1=1 \wedge 2=2$	Logical and
* &	∧	$\mathrm{d}x \wedge \mathrm{d}y$	Wedge product

Table 4.16. Homoglyphs supported by TeXmacs.

TeXmacs mainly relies on the standard variant system for entering homoglyphs. See Table 4.16 for the complete list of supported homoglyphs.

4.12 Customizing the semantics

We have done our best to support most of the traditional mathematical notations in TeXmacs. Nevertheless, you may sometimes need notations with a non-standard semantics. Certain areas may also require special notations that are not supported by default.

TeXmacs provides a very simple syntax primitive that allows you to manually override the default syntactical semantics of a formula. Assuming that semantic editing was activated, you may insert a syntax tag using ⌥x x or Insert ▸ Semantics ▸ Other. The first argument contains the formula as it should be displayed, whereas the second argument contains the formula as it should be interpreted.

For example, if we enter $\mathcal{R}$ as the first argument and $<$ as the second one, then the $\mathcal{R}$ will be interpreted as a binary relation, exactly in the same way as $<$. Moreover, the spacing around $\mathcal{R}$ will be adapted, so as to mimic the spacing around $<$. In this particular example, we might have obtained the same result by using the math-relation tag, which is essentially a syntax tag $<$ as second argument. Most standard operator types are available from the Insert ▸ Semantics menu or using the keyboard prefix ⌥x. In particular, you may use ⌥x ␣ to simply ignore a formula and ⌥x o in order to make the formula behave as an ordinary symbol (such as the letter "x").

The syntax primitive is particularly powerful in combination with the TeXmacs macro language. For instance, consider the formula $C=1/2\,\pi\,\mathrm{i}\oint f(z)\,\mathrm{d}z$. It is likely that the intended interpretation of $1/2\,\pi\,\mathrm{i}$ is $1/(2\,\pi\,\mathrm{i})$ and not $(1/2)\,\pi\,\mathrm{i}$. Therefore, if we often use the constant $2\,\pi\,\mathrm{i}$, then it is convenient to define a macro twopii by

```
⟨assign|twopii|⟨macro|⟨syntax|2 π i|(2 π i)⟩⟩⟩
```

You may group such macros together into style package with your favorite notations. Future versions of TeXmacs might also provide style packages with notations dedicated to specific areas.

Let us finally note that there are usually several ways for redefining the semantics of a formula. For example, an alternative way to define the macro twopii is using

```
⟨assign|twopii|⟨macro|2 π i⟩⟩
```

where we inserted a pair of invisible brackets around $2\,\pi\,\mathrm{i}$. Similarly, in the formula

$$\mathrm{e}^{\sqrt{x}+\mathrm{e}^{\sqrt{\log x}+\mathrm{e}^{\sqrt{\log\log x}+.\cdot^{\cdot^{\cdot}}\cdot.+\log\log\log x}+\log\log x}+\log x},$$

we may select the whole formula and give it the semantics of an ordinary symbol, by pressing ⌥x o. However, a nicer solution is to only select the subformula $.\cdot^{\cdot^{\cdot}}\cdot.$, and give it the semantics of an ordinary symbol. Yet another example is the sign sequence $++-+-+$ mentioned earlier. This sequence can be interpreted correctly by inserting invisible separators between the different signs using the , ␣ ␣ shortcut.

Chapter 5
Tables

Tables form one of the most versatile layout primitives of $\TeX_{\text{MACS}}$. You can create ordinary textual tables and specify the properties of individual cells in a very precise way. More generally, matrices, aligned lists of equations, and various other kinds of markup are also formatted as tables.

5.1 Different flavors of tables

The main kinds of tables are listed in the Insert ▸ Table menu and also available through the keyboard prefix ⌘t N. Here is an example of a "wide block" that was created using Insert ▸ Table ▸ Wide block or ⌘t N W:

Isaac Newton	1642–1726	English physicist and mathematician. With his book *Philosophiæ Naturalis Principia Mathematica* ("Mathematical Principles of Natural Philosophy"), he is considered to be the father of classical mechanics. Newton is also one of the founders of modern calculus.
Leonhard Euler	1707–1783	Prolific Swiss mathematician and physicist, who made major discoveries in many branches of mathematics and physics, such as infinitesimal calculus, number theory, graph theory, partial differential equations, fluid mechanics, astronomy, etc.

The distinctive feature of a "block" is that the borders of cells are visible as lines. Furthermore, a "wide" table extends over the entire width of a paragraph and long lines inside individual cells are wrapped when necessary.

Remark 5.1. In other word processors, "wide tables" are usually the default. $\TeX_{\text{MACS}}$ provides many kinds of tables that are *not* wide, in which case text inside cells is usually *not* line-wrapped. This is indeed most natural for small inline tables like matrices inside a formula. However, it is the responsibility of the user to prevent such tables from getting too large and not fitting on a single line. Line wrapping can be enabled via Focus ▸ Cell ▸ Line wrapping ▸ Top or by toggling the ☰ icon on the focus bar. In that case, you should also indicate the desired width of the cell (see section 5.3.2).

Insert ▸ Table ▸ Big table creates a "big" paragraph-wide table with a number and a legend. If you rather wish to fit several numbered tables with accompanying legends on a single line, then you may use Insert ▸ Table ▸ Small table. Tables 5.1 and 5.2 show an example of two small juxtaposed tables.

boom	tree
hallo	hello
wiskunde	mathematics

Table 5.1. Plain tabular, ⌘t N t.

boom	tree
hallo	hello
wiskunde	mathematics

Table 5.2. Centered block, ⌘t N B.

Inside math mode, a few additional kinds of tables Matrix, Determinant, Choice, and Stack are available through the Insert ▸ Table menu (see Figures 5.1–5.4). The stack tag corresponds to tables that are compressed as much as possible. This makes them useful inside subscripts, superscripts, or other places where little space is available.

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

Figure 5.1. Matrix

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

Figure 5.2. Determinant

$$f(x) = \begin{cases} 0 & \text{if } x \leqslant 0 \\ \mathrm{e}^{-\frac{1}{x}} & \text{if } x > 0 \end{cases}$$

Figure 5.3. Choice

$$\sigma_n = \sum_{\substack{1 \leqslant i \leqslant n \\ i \leqslant j \leqslant 2i}} \varphi_{i,j}$$

Figure 5.4. Stack

There are several other table-like environments and new ones may be created by the user. For instance, using Insert ▸ Mathematics ▸ Equations or ⌥&, you may insert an eqnarray* environment. This allows mathematical users to align lists of equations that extend over entire lines. An example of such a list of equations is

$$\begin{aligned} [\sin(f(x)\,g(x))]' &= (f(x)\,g(x))'\cos(f(x)\,g(x)) \\ &= (f'(x)\,g(x) + f(x)\,g'(x))\cos(f(x)\,g(x)) \end{aligned}$$

5.2 Basic table editing

When starting a new table, its size is minimal (usually 1×1) and its cells are empty. New rows and columns can be inserted using the keyboard shortcuts ⌥←, ⌥→, ⌥↑, and ⌥↓. For instance, ⌥↓ creates a new row below the current cursor position, as illustrated in Table 5.3 below. Alternatively, you may start a new row by hitting the ↩ key. New rows and columns can also be inserted from the Focus ▸ Resize menu, or using the icons ⊕, ⊕, ⊕, and ⊕ on the focus bar. Finally, it is possible to specify the exact number of rows and columns of a table using Focus ▸ Table ▸ Size ▸ Set number of rows and Focus ▸ Table ▸ Size ▸ Set number of columns.

	⌥→	⌥←	⌥↓	⌥↑	↩
$\begin{pmatrix} a & b\vert \\ c & d \end{pmatrix}$	$\begin{pmatrix} a & b & \vert \\ c & d & \end{pmatrix}$	$\begin{pmatrix} a & \vert & b \\ c & & d \end{pmatrix}$	$\begin{pmatrix} a & b \\ & \vert \\ c & d \end{pmatrix}$	$\begin{pmatrix} & \vert \\ a & b \\ c & d \end{pmatrix}$	$\begin{pmatrix} a & b \\ \vert & \\ c & d \end{pmatrix}$

Table 5.3. Inserting new rows and columns using the cursor keys. Starting from the initial 2×2 matrix at the left hand side, with the cursor behind the character b, the table indicates the results after pressing the keys ⌥→, ⌥←, ⌥↓, ⌥↑, and ↩.

	⌫	⌦
$\begin{pmatrix} a & \vert & b \\ c & & d \end{pmatrix}$	$\begin{pmatrix} a\vert & b \\ c & d \end{pmatrix}$	$\begin{pmatrix} a & \vert b \\ c & d \end{pmatrix}$
$\begin{pmatrix} a & b \\ \vert & \\ c & d \end{pmatrix}$	$\begin{pmatrix} a & b\vert \\ c & d \end{pmatrix}$	$\begin{pmatrix} a & b \\ \vert c & d \end{pmatrix}$

Table 5.4. Removing empty rows and columns using the keys ⌫ and ⌦.

Rows and columns that are entirely empty can be deleted using the keys ⌫ or ⌦, as indicated in Table 5.4. No matter whether they are empty or not, rows and columns can be deleted from the Focus ▸ Resize menu, or using the icons ⌫, ⌦, ⊠, and ⊠ from the focus bar.

Rows, columns, and more general rectangular selections of cells can easily be cut, copied, and pasted. Selections of cells are indicated using a different color (magenta) from that of usual selections (which are red). One typical scenario for copying and pasting a rectangular selection of cells is shown in Figure 5.5.

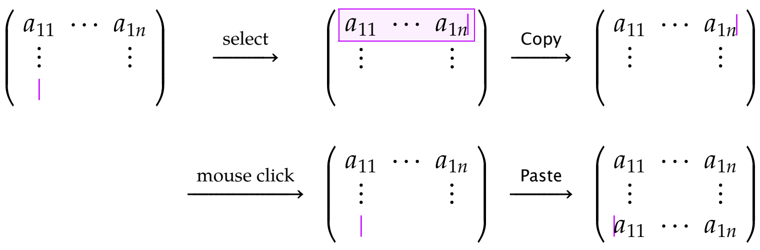


Figure 5.5. One typical scenario of copying and pasting a range of cells. Starting with the matrix at the left-hand side, we first select the top row using the mouse. We next copy the selection using Edit ▸ Copy or ⌘W and reposition the cursor inside the first cell of the third row. We finally paste the selection using Edit ▸ Paste or ^y.

5.3 Cell properties

5.3.1 Cell operation modes

The individual cells of tables have several properties that can be changed by the user: horizontal and vertical alignment, background color, width and height, border width and padding, etc. One first way to edit these properties is to open the cell property editor using Format ▸ Cell. Alternatively, changes can be made via the entries in the Focus ▸ Cell menu or the corresponding icons on the focus bar. Some properties can also be edited using the keyboard.

By default, cell operations are performed on the cell that contains the cursor. It is also possible to jointly modify the cell properties of all cells in the table, or of all cells in the row or column that contains the cursor. For this, it suffices to change the "cell operation mode", by selecting Entire table, Row or Column in the menu Focus ▸ Cell ▸ Operation mode. Similarly, if a selection of cells is active, then the properties are modified jointly for all cells in the selection.

Internally, cell properties are associated to rectangular regions of cells. Let us mention one useful implication of this fact. Assume that the first row of the following table was colored by selecting the first row and changing the background color to "pastel yellow" using Focus ▸ Cell ▸ Background color:

English	house	tree	beast
French	maison	arbre	bête
Dutch	huis	boom	beest

Then TEX_{MACS} associates this background color to the entire region corresponding to the first row. Now assume that we insert a new column between house and tree. This will yield

English	house		tree	beast
French	maison		arbre	bête
Dutch	huis		boom	beest

In other words, it is not necessary to specify the background color of the top cell again in the new column, since this cell is "naturally part of the region" for which we set the color to pastel yellow. This would not have been the case if we had separately specified the background color for each of the cells in the top row of the original table. In that case, the insertion of a new column between house and tree would have resulted in

English	house		tree	beast
French	maison		arbre	bête
Dutch	huis		boom	beest

5.3.2 Width and height

The widths of columns and heights of rows are computed automatically as a function of the content of the cells and constraints that can be specified by the user.

The width of a column is the maximum of the widths of its cells and similarly for the height of a row. The default width of a cell simply coincides with the width of its contents (except when you enable line wrapping, in which case

Nice	Dull
Even nicer	Even duller
Really very very nice	Extremely dull

Figure 5.6. Automatic width.

Nice	Dull
Even nicer	Even duller
Really very very nice	Extremely dull

Figure 5.7. Exact width of `1in`.

Nice	Dull
Even nicer	Even duller
Really very very nice	Extremely dull

Figure 5.8. Minimal width of `1in`.

Nice	Dull
Even nicer	Even duller
Really very very nice	Extremely dull

Figure 5.9. Maximal width of `1in`.

the default width becomes zero). The default width can be overridden using Focus ▸ Cell ▸ Width, Format ▸ Cell ▸ Width, or the icon menu on the focus bar. The user may either specify an exact, minimal, or maximal width for the cell.

When specifying a minimal width, the corresponding column has a guaranteed minimal width; the actual width may be larger if one of the cells is even wider. When specifying a maximal width for all cells, it is guaranteed that the width of the corresponding column will never exceed the specified value. However, this also means that any contents that don't fit into this width may overlap with neighboring cells. The same problem can occur if you specify an exact width. The possible settings are illustrated in Figures 5.6–5.9.

If your table is too wide or if you manually specified its width, then it may happen that the available horizontal space is not completely used up by the columns. This occurs for instance when starting a new wide table using Insert ▸ Table ▸ Wide block. By default, the unused space is distributed evenly over all columns, but this setting can be overridden by specifying stretch factors using Focus ▸ Cell ▸ Width ▸ Stretch factor or Format ▸ Cell ▸ Width ▸ Stretch. In Figure 5.10, we have shown an example table in which the unused space has been distributed evenly among the two columns. In Figure 5.11, we have shown the same table with a stretch factor of 0 for the first column and 1 for the second one.

The heights of rows are determined following similar principles. In addition, TeXmacs adjusts the vertical limits of cells in order to ensure a uniform layout for ordinary text. Indeed, characters like "a", "e", and "x" are shorter than "b", "d", and "k", and not as deep as "p", "q", and "g". If the vertical limits of cells

17	A Fermat prime number
60	Babylonian number base
256	Number of byte values

Figure 5.10. Evenly distribute unused space.

17	A Fermat prime number
60	Babylonian number base
256	Number of byte values

Figure 5.11. Grant all unused space to the last column.

were not adjusted, then the formula

$$\begin{pmatrix} a & x \\ x & a \end{pmatrix} + \begin{pmatrix} f & x \\ x & f \end{pmatrix}$$

would rather be displayed as follows:

$$\left(\begin{smallmatrix} a & x \\ x & a \end{smallmatrix}\right) + \begin{pmatrix} f & x \\ x & f \end{pmatrix}$$

In other words, TEXMACS adjusts cells such that they vertically extend to the highest and deepest character in the current font. This feature can be turned off, or only applied to the top or the bottom, via the menu Focus ▸ Cell ▸ Height ▸ Adjust limits or Format ▸ Cell ▸ Text height correction.

5.3.3 Borders and padding

The user may specify the widths of the four borders of each cell using Format ▸ Cell ▸ Border. The actual width of a border is taken to be the maximum of the specified border widths of adjacent cells. For instance, in the table below, all cells have a top and left border width of `2ln` and a bottom and right border width of `0ln`

top left	top	top right
left	central	right
bottom left	bottom	bottom right

The actual border width between the "top left" cell and the "top" cell is `2ln`, which is indeed the maximum of `0ln` (the right border width of the "top left" cell) and `2ln` (the left border width of the "top" cell). The length unit "`ln`" corresponds to the "standard line width" that is appropriate for the current font. In particular, the width of a fraction bar is `1ln`.

The specification of each of the borders of every individual cell can be tedious. For this reason, a simplified interface has been made available through Focus ▸ Cell ▸ Border and the icon on the focus bar. Using this interface, all four borders of a cell or selection of cells can be changed simultaneously, which includes the corresponding changes in the border widths of the neighboring cells. For example, after positioning the cursor inside the "central" cell of the above table, and clicking on , we obtain

top left	top	top right
left	central	right
bottom left	bottom	bottom right

Similarly, after selecting the four "central", "right", "bottom", and "bottom right" cells, and clicking on , we obtain

top left	top	top right
left	central	right
bottom left	bottom	bottom right

	0 sep	1 sep	1 spc	2 spc
0 sep	Test test	Test test	Test test	Test test
1 sep	Test test	Test test	Test test	Test test
1 spc	Test test	Test test	Test test	Test test
2 spc	Test test	Test test	Test test	Test test

Table 5.5. Typical settings for the horizontal and vertical cell padding.

Notice that the borders that are created in this way have a default width of `1 ln`. For other border widths, you may use ≡ ▸ Pen width or the more detailed interface from Format ▸ Cell.

Besides the borders, it is also possible to customize the amount of padding between borders of cells and their actual contents. Again, it is possible to separately specify the amount of padding in all four directions via Format ▸ Cell ▸ Padding, Focus ▸ Cell ▸ Padding, or the icon ≡. By default, TEXMACS uses a horizontal padding of `1 spc` (the default width of a space character for the current font) and a vertical padding of `1 sep` (standard separation space for the current font). In Table 5.5, we have illustrated the effect of various amounts of horizontal and vertical padding.

5.3.4 Horizontal and vertical alignment

The horizontal and vertical alignment of cells can be specified via Format ▸ Cell ▸ Alignment or Focus ▸ Cell ▸ Horizontal alignment and Vertical alignment. The current horizontal alignment is indicated on the focus bar by one of the icons ≡, ≡, ≡, or ⋮≡. The cell alignment can also be modified via the corresponding icon menu.

Besides the classic left, center, and right alignments, it is also possible to horizontally align on a decimal dot or comma. This is useful for numerical and monetary tables. Similarly, besides the classic top, center, and bottom alignments, cells can be aligned at their "baselines". This kind of vertical alignment is the default and usually indeed most appropriate for ordinary text. Table 5.6 shows the various available types of cell alignment.

	left	center	right	decimal dot	decimal comma
top	$x^2+y^2=z^2$	$\mathrm{e}^{\pi \mathrm{i}}=-1$	$H=\frac{1}{\frac{1}{x}+\frac{1}{y}}$	0.14285714	0,14285714
center	$H=\frac{1}{\frac{1}{x}+\frac{1}{y}}$	$x^2+y^2=z^2$	$\mathrm{e}^{\pi \mathrm{i}}=-1$	14.285714	14,285714
baseline	$\mathrm{e}^{\pi \mathrm{i}}=-1$	$H=\frac{1}{\frac{1}{x}+\frac{1}{y}}$	$x^2+y^2=z^2$	1428.5714	1428,5714
bottom	$\mathrm{e}^{\pi \mathrm{i}}=-1$	$H=\frac{1}{\frac{1}{x}+\frac{1}{y}}$	$x^2+y^2=z^2$	1428.5714	1428,5714

Table 5.6. Demonstration of the various kinds of cell alignment.

It is easy to modify the alignment of cells using the keyboard. The shortcut ⌘→ moves the horizontal alignment of a cell further to the right: a cell that was aligned at the left will be centered, and a cell that was centered will become right aligned. Similarly, the keystroke ⌘← moves the horizontal alignment further to the left, whereas ⌘↑ and ⌘↓ allow you to adjust the vertical alignment. When applied to a selection of cells, we notice that the selection remains active, so that multiple changes can be made efficiently. If you need to modify the horizontal alignment of many columns in multiple tables, then the keyboard shortcuts ⌘← and ⌘→ are also very efficient in combination with the "column operation mode" (Focus ▸ Cell ▸ Operation mode ▸ Columns).

5.3.5 Line-wrapping and block content

We already discussed the difference between wide tables and smaller inline tables such as matrices inside mathematical formulas. Wide tables allow cells to contain large portions of text. If the contents of a cell do not fit on a single line, then line-wrapping is activated by default in order to prevent the table from running out of the page. To this effect, we recall the existence of the toggle ≣ on the focus bar. Notice that line-wrapping only makes sense inside cells or tables for which a maximal width has been specified (see sections 5.3.2 and 5.4.2); otherwise, the cell will simply become as large as necessary for its contents to fit on a single line.

Line-wrapping can also be activated or deactivated using Format ▸ Cell ▸ Line wrapping or Focus ▸ Cell ▸ Line wrapping. These menus additionally allow you to specify the way in which line-wrapped cells should be aligned with respect to neighboring cells. For example, consider the following list of equations, created using Insert ▸ Mathematics ▸ Several equations:

$$\begin{array}{lcl}
a+b+c+d+e+f+g+h+i+j+k+ & & \\
l+m+n+o+p+q+r+s+t+u+v+ & & \\
w+x+y+z & = & z+y+x+w+v+u+t+s+r+q+p+ \\
 & & o+n+m+l+k+j+i+h+g+f+e+ \\
 & & d+c+b+a \\
 & = & a+z+b+y+c+x+d+w+e+v+f+ \\
 & & u+g+t+h+s+i+r+j+q+k+p+ \\
 & & l+o+m+n.
\end{array}$$

Both the first and the last column of this table are line-wrapped. However, the first column is aligned at the bottom whereas the last column is aligned at the top.

By default, cells in which line-wrapping is activated allow for "block content". Such cells may for instance contain several paragraphs, theorems, enumerations, or entire computer algebra sessions. In certain situations you may wish to allow for block content inside cells that are not line-wrapped. Inversely, you may want to disallow block content in cells where line wrapping is active. This can be specified via the menus Format ▸ Cell ▸ Block content or Focus ▸ Cell ▸ Block content.

5.4 Table properties

The global properties of a table can be edited after opening the "Table properties" editor via Format ▸ Table. Alternatively, you may use the submenus under Focus ▸ Table or the icon on the focus bar.

5.4.1 Number of rows and columns

We have already discussed the insertion and removal of rows and columns using the keyboard shortcuts ⌥←, ⌥→, ⌥↑, ⌥↓, or the icons , , , , , , , and on the focus bar. It is also possible to specify an exact number of rows or columns using Focus ▸ Table ▸ Size ▸ Set number of rows and Set number of columns. Moreover, the Focus ▸ Table ▸ Size menu allows you to specify a minimum or maximum number of rows or columns in a table. This is especially useful for the design of macros for customized tables (see section 5.7). For instance, the macro eqnarray* for aligned lists of equations

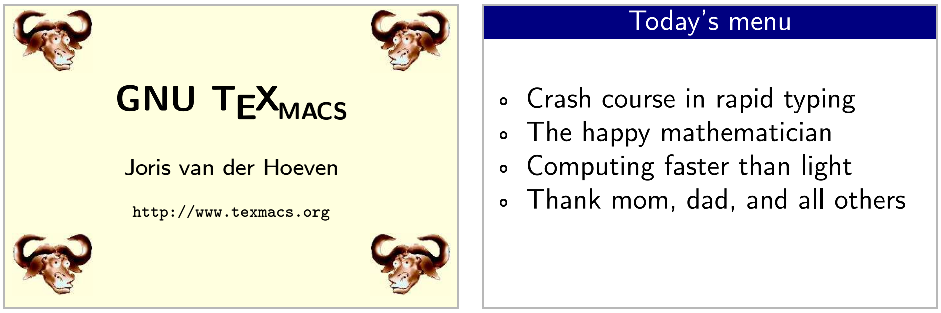
$$\begin{aligned} \mu &= \frac{a_1+\cdots+a_n}{n} \\ \sigma &= \sqrt{\frac{a_1^2+\cdots+a_n^2}{n}} \end{aligned}$$

specifies that "tables" of this kind should contain at least and at most three columns. Newly created lists of equations therefore contain three columns instead of one, but no new columns can be inserted using ⌥← or ⌥→. Similarly, the keys ⌫ and ⌦ suppress the table as soon as its cells are empty and there are only one row and three columns left.

5.4.2 Width and height

The width and the height of a table can be specified using Format ▸ Table ▸ Width and Height or Focus ▸ Table ▸ Width and Height, much in the same way as for cells (see section 5.3.2).

Two settings for the global table size are most common: wide tables on separate lines usually extend over the entire width of a paragraph, whereas the sizes of smaller tables such as matrices are typically determined automatically so as to make the contents of all cells fit. Toggling between these two possibilities is so common that the focus bar contains a special icon for this purpose. You may also use Focus ▸ Table ▸ Width ▸ Paragraph or specify `1par` as Format ▸ Table ▸ Width or Focus ▸ Table ▸ Width.



GNU TEXMACS

Joris van der Hoeven

http://www.texmacs.org

Today's menu

- Crash course in rapid typing
- The happy mathematician
- Computing faster than light
- Thank mom, dad, and all others

Figure 5.12. Example of two miniscreens for a presentation.

Of course, other table sizes can sometimes be useful. For instance, one may use a fixed table size for mimicking a minipage or a miniscreen, as in Figure 5.12.

5.4.3 Borders and padding

You may specify the width of the left, right, top, and bottom borders of a table using Format ▸ Table ▸ Border or Focus ▸ Table ▸ Border. This feature has been provided for consistency, but is really redundant with the possibility to specify the borders of selections of cells (see section 5.3.3): it suffices to select all cells. On the other hand, padding is applied around the table, instead of inside the cells. You may change the padding at the left, right, top, and bottom of the table using Format ▸ Table ▸ Padding or Focus ▸ Table ▸ Padding.

5.4.4 Alignment with respect to the surrounding text

The global alignment properties of a table specify how to align it with respect to the surrounding text. These properties can be modified using Format ▸ Table ▸ Alignment or Focus ▸ Table ▸ Horizontal alignment and Vertical alignment. Customized alignments are mainly needed for "inline" tables, which are embedded into a formula or paragraph with other text.

By default, tables are vertically aligned at the *axis*. The axis is located halfway between the top and the bottom of the character "x". Tables that are centered along the axis roughly descend as much below the bottom of the text as they exceed the top. When selecting Top, Middle or Bottom for the vertical alignment, the top, center, or top of the table will instead be aligned with the *baseline* of the surrounding text: see Figure 5.13.

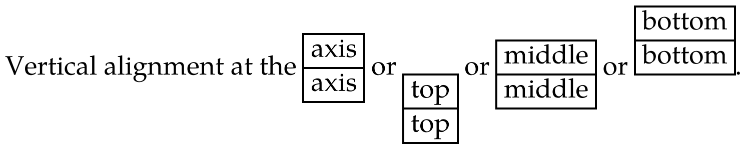


Figure 5.13. Basic types of vertical table alignment with respect to surrounding text.

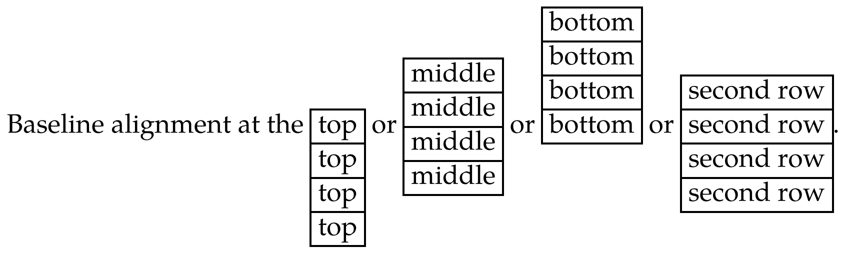


Figure 5.14. Various kinds of vertical baseline alignment of tables.

It is also possible to align the baseline of a particular row in the table with the baseline of the surrounding text, by selecting Top baseline, Middle baseline, Bottom baseline or Specific baseline in Focus ▸ Table ▸ Vertical alignment. This kind of alignment is illustrated in Figure 5.14. When the number of rows of a table is even, the middle baseline is located exactly at the middle between the baselines of the two central rows. For the rightmost table in Figure 5.14, we chose "2" for Focus ▸ Table ▸ Vertical alignment ▸ Specific baseline.

The default horizontal alignment of tables is at the left. Other alignments are supported for completeness, but rarely needed, since they usually result in overlapping text: see Figure 5.15.

5.4.5 Breaking up large tables

Unfortunately, $\mathrm{T_EX_{MACS}}$ is currently not very efficient for editing large tables. In particular, it is recommended to manually break up tables that do not fit on a single page. Smaller tables sometimes do not nicely fit on a page either. This typically happens for long lists of equations such as

$$\begin{aligned}
x_1 &= \Phi_1(x_1,x_2,\ldots,x_{n-1},x_n)\\
x_2 &= \Phi_2(x_1,x_2,\ldots,x_{n-1},x_n)\\
&\vdots\\
x_{n-1} &= \Phi_{n-1}(x_1,x_2,\ldots,x_{n-1},x_n)\\
x_n &= \Phi_n(x_1,x_2,\ldots,x_{n-1},x_n).
\end{aligned}$$

If you want to allow for page breaks inside such tables, then you should use Format ▸ Table ▸ Large tables ▸ Enable page breaking or Focus ▸ Table ▸ Special ▸ Table breaking. Notice that you should set all interline and interparagraph spacing to zero (see section 3.9) whenever the cells of such a table admit borders.

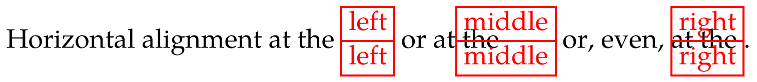


Figure 5.15. Horizontal table alignment with respect to surrounding text.

		2011		2012		Total	
		jan–jun	jul–dec	jan–jun	jul–dec		
Beverages	Milk	120.10	116.20	128	126.15	490.45	850.65
	Cola	85.40	105.30	74.15	95.35	360.20	
Fruit	Apples	65.10	72.85	82.90	83.10	303.95	728.40
	Cherries	34.20	55.45	23.85	57.55	171.05	
	Bananas	83	47.20	72	51.20	253.40	
Total		387.80	397	380.90	413.35	1579.05	

Table 5.7. Spending table that illustrates how to use joined cells.

5.5 Joined cells and subtables

It is sometimes useful to merge several cells into a single cell. After making a selection of cells, this can be done by clicking on the icon or Focus ▸ Cell ▸ Join selected cells. Cells that have been joined together can be dissociated using or Focus ▸ Cell ▸ Dissociate joined cells. Table 5.7 gives a typical example of a table with a few joined cells. Cell-joining can also be an efficient way to create certain diagrams: see Figure 5.16. In this example the widths and heights of all cells were set to `0.25in`.

Remark 5.2. When joining selected cells $C_{11}, \ldots, C_{kl}$, the top left cell C_{11} will assume the role of the "leader" and contains the content of the joined cell. The other cells $C_{12}, \ldots, C_{kl}$ remain part of the table, but become invisible to the user until the joined cell is dissociated.

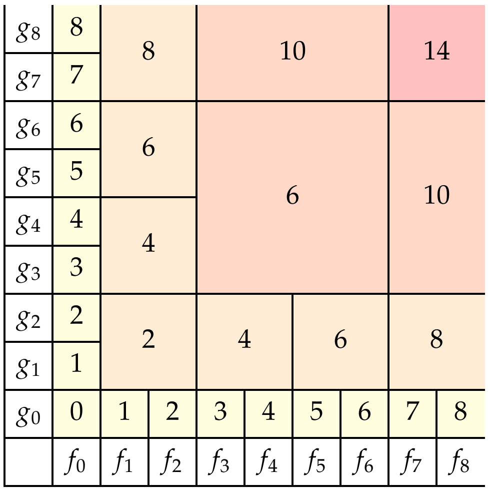


Figure 5.16. Fast relaxed multiplication of formal power series.

Start with	a row
Next one	
End	with a row

Start with	a row
Next one	
End	with a row

Figure 5.17. The left-hand side shows a table with three rows and one column in which we created two subtables in the first and the last row. The right-hand side shows a table with three rows and two columns in which we joined the two cells in the middle row. Notice the differences in the alignment.

Conversely, TeXmacs also allows you to split up a single cell into several ones, by transforming it into a "subtable". This can be done using the ▦ icon on the focus bar or Focus ▸ Insert subtable. Whether it is better to join several cells together or split up some of the other cells depends on the context. The main difference between the two techniques is that the alignment properties of cells in subtables of the outer table are not correlated: see Figure 5.17.

5.6 Artistic tables

Nice graphical effects can be obtained through the suitable use of colors and fonts inside tables. Let us illustrate this with a few examples. In Table 5.8, we gave the first row a dark blue background color and a pastel yellow foreground color. In addition, we used a "sans serif" font for the text, which enhances the readability. Notice that the usual ways to change the foreground color (Format ▸ Color or the ✎ icon) and font (Format ▸ Font) also apply for selections of cells. On the other hand, this mechanism does not work for content tags from the Insert menu such as strong or name.

Dutch	English	French
huisje	little house	petite maison
boompje	little tree	petit arbre
beestje	little beast	petite bête

Table 5.8. Example of table in which the first row uses light colors on a dark background.

Tables can also be given a more artistic look through the use of background patterns. One example of this technique is shown in Table 5.9. The "wooden borders" are empty rows and columns whose heights and widths were respectively set to `0.5em`. We also turned off the height correction using Focus ▸ Cell ▸ Height ▸ Ajust limits ▸ None.

Dot hatches	Line hatches	Floral	Fashion
Light	Single	Grey floral	Leather
Medium	Double	Symphony	Feathers
Dark	Triple	Regal	Textile

Table 5.9. The use of background patterns inside tables.

```
forall (K: Field)
power (x: K, n: Int): K == {
  if n > 1 then {
    p: K == square power (x, n quo 2);
    if n rem 2 = 0 then return p;
    else return x * p;
  }
  else if n = 1 then return x;
  else if n = 0 then return 1;
  else return invert power (x, -n);
}
```

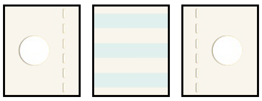


Table 5.10. Combining the three above background patterns inside a table.

Using a combination of background patterns, you may also create various kinds of "artwork paper". For instance, in Table 5.10, we used three different background patterns: two for the left and right borders, and another one for the "paper". Using six additional patterns for the top and bottom borders, it is possible to design even more elaborate decorations.

5.7 Table macros

Whenever a particular kind of table style is needed at many places in a document, it may be worth it to create a "*table macro*". For detailed information about the creation of macros, we refer to chapter 12.

Let us explain the creation of a simple table macro with an example. The first step is to create a prototype of the table with the style that you are interested in. Assume for instance that we use the following table for our prototype:

Dutch	English	French
huisje	little house	petite maison
boompje	little tree	petit arbre
beestje	little beast	petite bête

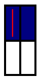
It may be useful to specify a minimum size as well using Focus ▸ Table ▸ Size ▸ Minimal number of rows and Minimal number of columns. In our example, it is natural to require at least two rows and two columns.

At the second step, you should place the cursor anywhere inside the table an click on Tools ▸ Macros ▸ Create table macro. A window for editing macros should pop up with a declaration that should look as follows:

```
| :=
⟨macro|body|
    ⟨tformat|
        ⟨cwith|1|-1|1|-1|cell-rborder|1ln⟩|
        ⟨cwith|1|-1|1|-1|cell-bborder|1ln⟩|
        ⟨cwith|1|1|1|-1|cell-tborder|1ln⟩|
        ⟨cwith|1|-1|1|1|cell-lborder|1ln⟩|
        ⟨cwith|1|1|1|-1|cell-background|dark blue⟩|
        ⟨cwith|1|1|1|-1|font-family|ss⟩|
        ⟨cwith|1|1|1|-1|color|pastel yellow⟩|
        ⟨twith|table-min-rows|2⟩|
        ⟨twith|table-min-cols|2⟩|
        body⟩⟩
```

At the default cursor position, enter the name of your macro and click on the Ok button.

Having created our table macro, we may now use it. Assuming that we called it "translation-table", it suffices to type \ followed by this name and hit ↩. This will yield



Notice that the table macro only preserves the style of the macro, not its actual contents. Notice also that the initial table macro application indeed contains two rows and two columns, as desired.

In certain cases, we might want to surround the actual table with some fixed content. For instance, a matrix is really a table surrounded by two big parentheses. Assume that we want to create a new curly-matrix macro for which the parentheses are replaced by curly braces. As usual, we first create a prototype of such a "table":

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

Next, we carefully select the table together with the braces and click on Tools ▸ Macros ▸ Create table macro. As before, we finally enter the name curly-matrix in the macro editor and click on the Ok button. Notice that the selection step is essential: if we just place our cursor inside the table, then Tools ▸ Macros ▸ Create table macro creates a macro for the table without the surrounding braces.

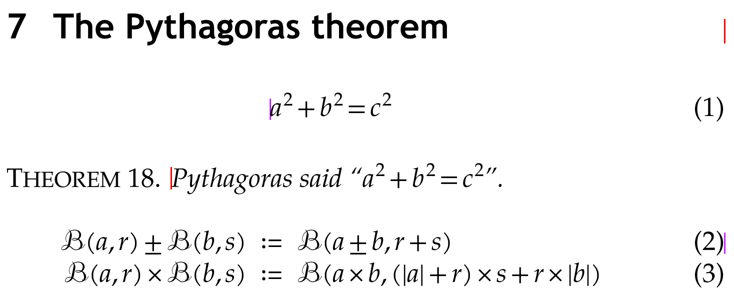
Chapter 6
Connected documents

The days are gone that we had to roam through the alleys of libraries, when brinks of information were hiding themselves inside thick books. Modern documents need to be *connected*, within ten-click distance of the entire universe.

TEXMACS provides functionalities for the creation of many types of connections inside and between documents. In this chapter, we cover hyperlinks, labels and references, tables of contents, indexes, bibliographies, inclusions from other documents, etc. In chapter 11, we will explain how TEXMACS can be connected to external computational software. Other types of "connections" are under development, such as support for various web services.

Connected documents contain a lot of automatically generated content, ranging from short references to complete bibliographies. This introduces an inevitable pitfall: the precise way in which this content has to be generated and kept up to date may become a complex and potentially time-consuming process. In order to avoid any slowdown of the usual editing process, some of these updates are only done on request using Document ▸ Update ▸ All or by clicking on the icon. The Document ▸ Update menu also contains several entries for specific kinds of updates, such as regenerating only the table of contents. Note that you have to be in "paper mode" (Document ▸ Page ▸ Format ▸ Page rendering ▸ paper) if you want page numbers to be updated correctly.

The preceding discussion explains why the updating process of connected documents is only semi-automatic. Of course, TEXMACS attempts to automate things as much as possible: updates that can be performed sufficiently quickly are done on-the-fly. As computers get faster, and with the implementation of more efficient algorithms inside the editor, time works in favor of more and more automation. However, efficiency is not the only reason for keeping the updating process semi-automatic: some types of automatically generated content may never reach a "stable state". For instance, a tag that displays the current time is bound to change at every update. Similarly, some updates might involve computations by computer algebra systems, with potentially different answers at every new execution. To a lesser extent, this phenomenon may occur when updating tables of contents, bibliographies, etc. Indeed, the regenerated table of contents might require an extra page, in which case an additional update is needed to get all page numbers right. It is therefore recommended to systematically update important documents a few times before going public.


7 The Pythagoras theorem

$$a^2+b^2=c^2 \tag{1}$$

THEOREM 18. *Pythagoras said "$a^2+b^2=c^2$".*

$$\begin{aligned} \mathcal{B}(a,r)\pm\mathcal{B}(b,s) &:= \mathcal{B}(a\pm b,r+s) \qquad (2)\\ \mathcal{B}(a,r)\times\mathcal{B}(b,s) &:= \mathcal{B}(a\times b,(|a|+r)\times s+r\times|b|) \qquad (3)\end{aligned}$$

Figure 6.1. Appropriate cursor positions for where to insert labels.

6.1 Labels and references

In order to create references to sections, equations, theorems, etc., you must proceed in two steps. You first have to create a *label* at the position to which you want to refer. For equations, the label should be created inside the equation, and similarly for theorems and most other tags. The labels for sections, subsections, etc. should rather be positioned just after the section titles; this prevents TeXmacs from duplicating the labels in the table of contents. When labeling multiple equations (that were created using Insert ▸ Mathematics ▸ Equations), you must put the labels just behind the equation numbers (which can be toggled using ^#). The appropriate positions for labels are illustrated in Figure 6.1.

You may create the actual label using Insert ▸ Link ▸ Label or the keyboard shortcut ⌘!. Newly created labels are initially inactive: you are supposed to enter the name of your label and then hit ↩ in order to activate it. After that, a small marker should appear to indicate the position of the label. This marker will not appear in the printed version of your document. When selecting Document ▸ Informative flags ▸ Detailed, these markers will also indicate the names of the actual labels, as in the equation

$$\overset{\text{eq:Pythagoras}}{a^2+b^2=c^2}. \tag{6.4}$$

It is the responsibility of the user to avoid the creation of several labels with the same name. In that case, you will receive an error message, since references to such labels are ambiguous.

Assuming that you created a label `test-name`, you can insert a *reference* to it using Insert ▸ Link ▸ Reference or the keyboard shortcut ⌘?. Again, a newly created reference will be inactive. After re-entering the name `test-name` and hitting the activation key ↩, the appropriate reference should appear in the text. For equations such as (6.4), we note that you have to add the surrounding parentheses yourself. When entering the name of a reference, you may benefit from the mechanism of "tab-completion": after typing the first characters of the name, hitting the ⇥ key will automatically complete it. References to non-existent labels cause $\mathrm{T_EX_{MACS}}$ to complain with an error message. Inside the editor, as well as for the PDF and HTML exports, references behave like hyperlinks: when clicking on them, you jump to the corresponding label.

6.2 Hyperlinks and actions

It is possible to create hyperlinks (similar to the ones found on web pages) using ⌘i > or Insert ▸ Link ▸ Hyperlink. The first field of the hyperlink contains the visible text. The second field specifies the location you wish to point at; this may be the URL of a web page like `http://www.texmacs.org` or the name of a local file. After activation using ↩, hyperlinks are rendered using a blue-grey color on the screen and without any special color in printed documents. The screen color changes to purple-grey as soon as you click on the link. The coloring schemes can be customized using Focus ▸ Preferences ▸ Locus color and Focus ▸ Preferences ▸ Visited color. As is usual for hyperlinks on the web, a link of the form *`#label`* points to a label in the same document and a link of the form *`url#label`* points to a label in the document located at *`url`*.

Instead of jumping to a specified location, it is also possible to execute arbitrary commands when clicking on a "link". This is done by creating an "*action*" using ⌘i * or Insert ▸ Link ▸ Action. The first field again contains the text to be displayed, whereas the second field now specifies a Guile/Scheme command. After activation, the Scheme script will be executed each time you click on the text. For security reasons, such scripts are not always accepted. By default, you are prompted for acceptance; this default behavior may be changed in Edit ▸ Preferences ▸ Other ▸ Security. Notice that the Guile/Scheme command

```
(system "shell-command")
```

evaluates `shell-command` as a shell command. This provides an easy way to launch any external application via a simple mouse click.

6.3 Inserting images

You can insert images into the text using Insert ▸ Image. The entries Insert ▸ Image ▸ Draw image and Insert ▸ Image ▸ Draw over selection allow you to draw your own pictures using the graphical editor that comes with $\mathrm{T_EX_{MACS}}$ (see chapter 8).

External images can be incorporated using either Insert ▸ Image ▸ Link image or Insert ▸ Image ▸ Insert image. The first method only puts a link to the image inside your document, whereas the second method inserts the image itself. Links offer the advantages that documents remain smaller in size, that the same image can be shared between several documents or document fragments, and that images can be updated independently from documents. The main drawback is that documents become less self-contained: if you send your document to a friend or colleague, then you also need to send each of the linked images. This drawback can be alleviated by keeping all images inside a dedicated folder.

By default, external images are displayed at their design sizes and aligned at their bottom lines. Alternative widths, heights, and alignment offsets can be specified using the focus menu or toolbar.

- When specifying a new width (using Focus ▸ Set width), but no height (or *vice versa*), the image is resized so as to preserve the aspect ratio. For example, entering a width of `1par` will make the image extend over the entire paragraph width and adjust the height proportionally.

  You may use `w` and `h` as special lengths for the default width and height of the image. For example, specifying `2w` and `2h` for the width and the height, the image will be displayed at twice the default size.

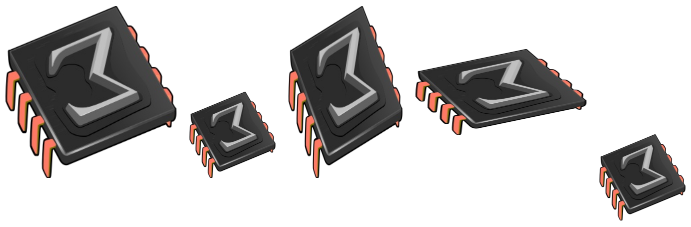
- When specifying an alternative offset (using Focus ▸ Set x-offset and Focus ▸ Set y-offset), you may use the `w` and `h` lengths for the displayed width and height (i.e. `w` and `h` no longer stand for the default width and height). For example, using `-0.5h` for the *y*-offset will vertically center the image.

Figure 6.2 shows the effect of various size and offset settings for a given image.

In addition to the above ways to specify the exact width, height, and offset of an image, it is also possible to customize these settings efficiently using the keyboard: the shortcuts ⌘← and ⌘→ (resp. ⌘↑ and ⌘↓) allow you to decrease and increase the width (resp. height), whereas ⌘⇟ and ⌘⇞ can be used for adjusting the vertical offset.

Currently, TEXMACS supports the following image formats:

`bmp`	Windows Bitmap
`eps`	Encapsulated Postscript
`fig`	Figures for the XFIG program
`gif`	Graphic Interchange Format
`jpg`, `jpeg`	Joint Photographic Experts Group
`png`	Portable Network Graphics
`pbm`	Portable Bitmap
`pdf`	Portable Document Format
`ppm`	Portable Pixmap
`ps`	Postscript
`tiff`	Tagged Image File Format
`xpm`	X11 Pixmap



Width		0.5w	15mm	1w	
Height			1h	0.5h	0.5h
X-offset					
Y-offset				0.5h	-0.5h

Figure 6.2. Example of the effect of various size and offset specifications for an image. Blank fields correspond to the default setting. In particular, the leftmost image corresponds to the default rendering.

The POSTSCRIPT support relies on GHOSTSCRIPT. Binary TeXmacs distributions include GHOSTSCRIPT, but you may need to install GHOSTSCRIPT yourself if you compile TeXmacs from the source code.

More image formats such as `svg` (Scalable Vector Graphics) are supported by TeXmacs modulo the installation of appropriate converters on your system [10]. On UNIX systems, we recommend the installation of IMAGEMAGICK, which comes with a large collection of converters [8].

6.4 Content extraction

6.4.1 Tables of contents

In order to generate a table of contents for your document, you should proceed as follows:

1. Put the cursor at the appropriate place and click on Insert ▸ Automatic ▸ Table of contents.

2. Select Document ▸ Page ▸ Format ▸ Page rendering ▸ paper for the page mode.

3. Generate the table of contents using Document ▸ Update ▸ Table of contents or Document ▸ Update ▸ All.

As explained in the introduction of this chapter, the second step is necessary in order to get the page numbers right. For the final version of a document, we also recall that it is advisable to update it several times, until a stable state is reached. Indeed, the page numbers may change as a result of modifications in the table of contents!

```
⟨index|simple⟩
⟨subindex|entry|subentry⟩
⟨subsubindex|entry|subentry|subsubentry⟩
⟨index-complex||strong||⟨tuple|entry|important⟩⟩
⟨index-complex|theory|||⟨tuple|α-theory⟩⟩
⟨index-complex|||range|⟨tuple|entry|range⟩⟩
⟨index-complex|||range|⟨tuple|entry|range⟩⟩
⟨index-line|e|⟨strong|E⟩⟩
⟨index-line|s|⟨vspace*|0.5fn⟩⟨strong|S⟩⟩
⟨index-line|t|⟨vspace*|0.5fn⟩⟨strong|T⟩⟩
```

Index

E

entry
- important **109**
- range 109–109
- subentry 109
  - subsubentry 109

S

simple 109

T

α-theory 109

Figure 6.3. Demonstration of the available types of index entries. The left-hand side shows the entries in their deactivated state. The right-hand side shows the index generated from these entries, after activating them and putting them at appropriate places in the main text.

6.4.2 Indexes

For the generation of an index, you first have to put "index entries" in your document using Insert ▸ Link ▸ Index entry. In a second step, you must put your cursor at the place where you want your index to be generated and click on Insert ▸ Automatic ▸ Index. The index can then be generated in a similar way as the table of contents (see Figure 6.3 for an example).

In the Insert ▸ Link ▸ Index entry menu, you find several types of index entries. The simplest ones are Main, Sub, and Subsub, which are macros with one, two, and three arguments respectively. Entries of the form Sub and Subsub may be used to subordinate index entries with respect to other ones. As usual, you have to fill out the arguments of the entries, and then hit ↩ in order to activate them. The locations of index entries are indicated through markers or flags, as in the case of labels (see section 6.1).

A Complex index entry takes four arguments. The first one is a key that specifies how the entry has to be sorted; it must be a "tuple" (created using ⌘i <) whose first component is the main category, the second a subcategory, etc. The second argument of a complex index entry is either blank or "strong", in which case the page number of your entry will appear in bold. The third argument is usually blank, but if you create two index entries with the same non-blank third argument, then this will create a "range" of page numbers. The fourth argument, which is again a tuple, is the entry itself.

It is also possible to create an index line without a page number using Insert ▸ Link ▸ Index entry ▸ Interject. The first argument of this macro is a key for how to sort the index line. The second argument contains the actual text. This construct may be useful for creating different sections "A", "B", etc. in your index.

```
⟨glossary-line|⟨strong|Symbols⟩⟩
⟨glossary|ℵ⟩
⟨glossary-explain|ℝ|The field of real numbers⟩
⟨glossary-line|⟨vspace*|0.5fn⟩⟨strong|Notations⟩⟩
⟨glossary-explain|A∐B|Coproduct of A and B⟩
⟨glossary-dup|A∐B⟩
```

Glossary

Symbols

$\aleph$		110
$\mathbb{R}$	The field of real numbers	110

Notations

$A \amalg B$	Disjoint union of A and B	110, 110

Figure 6.4. Demonstration of the available types of glossary entries. The left-hand side shows the entries in their deactivated state. The right-hand side shows the glossary generated from these entries.

6.4.3 Glossaries

Glossaries are compiled in a similar way as indexes, but the entries are not sorted (see Figure 6.4 for an example): the items in the glossary are listed in the same order as the corresponding entries in the text. Glossary entries are created using Insert ▸ Link ▸ Glossary entry, whereas the glossary itself is inserted using Insert ▸ Automatic ▸ Glossary.

A Regular glossary entry just contains some text, and a page number will be generated for it. An Explained glossary entry contains a second argument, which explains the notation. A Duplicate entry may be used to create a page number for the second occurrence of an entry. A glossary line creates an entry without a page number.

6.4.4 Lists of figures and tables

Using Insert ▸ Image ▸ Big figure or Insert ▸ Image ▸ Small figure, it is possible to associate a number and a *caption* (i.e. some explicative text) to a figure. TeXmacs allows you to automatically produce the list of all figures of this kind: for each figure, this list contains its caption and page number. Such a list of figures is inserted using Insert ▸ Automatic ▸ List of figures and generated using Document ▸ Update ▸ List of figures or Document ▸ Update ▸ All.

Sometimes, the caption of a figure may be quite long. In that case, you may wish to use an alternative shorter caption for the list of figures. This can be done by using the tag caption-detailed as your caption. The first argument of this tag should be the long caption, and the second argument the shorter version.

The same mechanism applies for numbered tables with captions, as created using Insert ▸ Table ▸ Big table or Insert ▸ Table ▸ Small table. Lists of tables are inserted using Insert ▸ Automatic ▸ List of tables and the tag caption-detailed can again be used for generating alternative short captions in such lists.

```
⟨index|power series⟩
⟨with-index|hof|⟨index|Isaac Newton⟩⟩
⟨subindex|relativity|special⟩
⟨subindex|relativity|general⟩
⟨with-index|hof|⟨index|Albert Einstein⟩⟩
⟨index|Hilbert scheme⟩
⟨with-index|hof|⟨index|Alexander Grothendieck⟩⟩
```

Index

Hilbert scheme 111
power series 111
relativity
    general 111
    special 111

Hall of fame

Albert Einstein 111
Alexander Grothendieck 111
Isaac Newton 111

Figure 6.5. Example of how to create two indexes.

6.4.5 Multiple extractions

In a thick book or a collection of papers, you sometimes may wish to include a separate table of contents or index for each chapter, together or not with one for the entire book. Similarly, you may wish to maintain separate indexes for concepts and people. To this effect, you should first know that each automatically generated section comes with an identifier: by default, $\mathrm{T_EX_{MACS}}$ uses `toc` for the table of contents, `bib` for the bibliography, `idx` for the index, `gly` for the glossary, `lof` for the list of figures, and `lot` for the list of tables. For selected fragments of text, you may specify alternate identifiers with Insert ▸ Link ▸ Alternate, and use this to generate a separate section for each identifier.

Let us show with an example how this works. Assume that we wish to maintain two separate indexes `idx` and `hof` for general concepts and names of famous persons. The first index is managed as usual. The second one can be created as follows: click on Insert ▸ Link ▸ Alternate ▸ Index and specify `hof` at the prompt; next insert the index as usual via Insert ▸ Automatic ▸ Index. The name “Hall of fame” can be specified by selecting the new index and doing Focus ▸ Rename. Index entries for persons are inserted in a similar way: click on Insert ▸ Link ▸ Alternate ▸ Index and specify `hof` at the prompt; next insert the index entry as usual via Insert ▸ Link ▸ Index entry. An example is shown in Figure 6.5.

The specification of an alternate identifier for each person in the hall of fame may be somewhat tedious. Of course, nothing prevents you from creating a macro hof-index for this purpose:

```
⟨assign|hof-index|⟨macro|name|⟨with-index|hof|⟨index|name⟩⟩⟩⟩
```

Similar macros can be created for putting the same entry in several indexes. See chapter 12 for more information on how to create your own macros.

6.5 Bibliographies

The compilation of a bibliography is similar to the generation of an index (as described in section 6.4.2), with "citations" in the role of "index entries". There is one major difference: the bibliographic information is usually stored in a separate database. This makes it easy to share such information among different documents. The "bibliographic database" can either be maintained by hand inside a file, or with the help of TeXmacs in the form of a genuine database.

TeXmacs essentially uses BibTeX [43] as the format for storing bibliographic entries. In particular, users who are familiar with this format may directly use existing or hand-compiled BibTeX files as bibliographic databases. There are also several external tools for managing BibTeX-based databases, such as BibDesk [57] and Zotero [50]. Some of these tools make it easy to retrieve existing bibliographic entries from the web. Section 6.5.4 contains a crash course on BibTeX; more information can be found in [43, 38, 16].

The particular BibTeX file with your bibliographic database can be specified when inserting a bibliography into your document. It is also possible to let TeXmacs do the job of maintaining a global bibliographic database for all your documents. This requires you to enable the Database tool in the Tools menu. One advantage is that you may directly perform searches from inside the editor. Another advantage is that relevant entries are copied as invisible attachments inside your documents. When collaborating on a document, this makes it possible to automatically share bibliographic data among colleagues. The built-in bibliography manager still allows you to import and export bibliographic data in the form of BibTeX entries or files.

6.5.1 Importing bibliographic references

A typical BibTeX file `example.bib` with three entries is shown below. We will use it for all our examples in this section. Bibliographic entries for many classical works can be found on the web in BibTeX format. More generally, BibTeX files can be assembled and edited by hand using TeXmacs or any external ASCII text editor.

In order to import the example file into the integrated bibliography manager (assuming that Tools ▸ Database tool is enabled), you first have to open your personal bibliography database using Data ▸ Open bibliography. It next suffices to click on Data ▸ Import and select `example.bib` using the file browser.

example.bib

```
@Article{Ein05,
  author =       {Einstein, Albert},
  title =        {Ist die {Tr\"agheit} eines {K\"orpers}
                  von seinem {Energieinhalt} abh\"angig?},
  journal =      {Annalen der Physik},
  year =         {1905},
  month =        {March},
  number =       {113},
  volume =       {323},
  pages =        {639--641}
}

@Book{New1671,
  author =       {I. Newton},
  title =        {De methodis serierum et Fluxionum},
  publisher =    {Manuscript},
  year =         {1671}
}

@Article{Tur36,
  author =       {A. Turing},
  title =        {On computable numbers, with an application
                  to the {Entscheidungsproblem}},
  journal =      {Proc. London Maths. Soc.},
  year =         {1936},
  volume =       {2},
  number =       {42},
  pages =        {230--265}
}
```

6.5.2 Inserting citations

Ordinary citations such as [2] are created using Insert ▸ Link ▸ Citation ▸ Visible. The entries in the database are identified via their keys. In our example file `example.bib`, the keys are `New1671`, `Ein05`, and `Tur36`. Using Focus ▸ Insert argument after or ⊞, further citations can be added to form a group. For instance, the three entries from our example file are [1, 2, 3].

If you forgot part of a key, then you may use tab-completion to cycle through all possible completions. You may also search for bibliographic entries in your personal database using Focus ▸ Search in database or by clicking on the 🔍 icon on the focus bar. After entering some keywords separated by spaces in the Search field of the search window, all entries are listed in which each keyword appears at least once. By clicking on the key of one of the matching entries, this key will be chosen for your citation.

You sometimes may wish to mention a reference in the bibliography, without putting any corresponding citation in the main text. In that case, you have to create an "invisible citation" using Insert ▸ Link ▸ Citation ▸ Invisible.

Finally, Insert ▸ Link ▸ Citation ▸ Detailed allows you to cite a specific portion of text in a work, such as a section or a theorem. For example, see [3, section 11] for a proof of the impossibility to solve the *Entscheidungsproblem*.

Bibliography

[1] Albert Einstein. Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik*, 323(113):639–641, March 1905.

[2] I. Newton. *De methodis serierum et Fluxionum*. Manuscript, 1671.

[3] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.

6.5.3 Compiling the bibliography

You may insert the bibliography using Insert ▸ Automatic ▸ Bibliography. If you enabled Tools ▸ Database tool, then your personal global database will be used for the bibliographic data. Otherwise you will be prompted to select a BIBTEX file using the file browser. With the focus on the bibliography, the file name can be changed *a posteriori* using Focus ▸ Set file name or on the focus toolbar. An empty file name instructs TEXMACS to search for bibliographic entries in your personal database.

The BIBTEX style to be used can also be specified on the focus toolbar or using Focus ▸ Style, just after insertion of the bibliography. Notice that the Style text on the focus toolbar is actually a button that opens a popup menu with some of the supported styles. TEXMACS contains built-in support for the most frequent BIBTEX styles `abbrv`, `acm`, `alpha`, `elsart-num`, `ieeetr`, `plain`, `siam`, `unsrt`. In order to generate bibliographies using this built-in support, you must prefix the bibliography style with `tm-`. For instance, the default style `tm-plain` corresponds to `plain`. Native BIBTEX styles can also be used, modulo installation of the `bibtex` program on your computer.

The bibliography styles determine the rendering of both citations and the entries in the bibliography itself, as well as the way these entries are sorted. Below, we have shown the effect of the standard styles for our sample bibliography `example.bib`.

`tm-abbrv`

The achievements in [2, 1, 3] have changed the world.

[1] A. Einstein. Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik*, 323(113):639–641, March 1905.

[2] I. Newton. *De methodis serierum et Fluxionum*. Manuscript, 1671.

[3] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.

`tm-acm`

The achievements in [2, 1, 3] have changed the world.

[1] EINSTEIN, A. Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik 323*, 113 (March 1905), 639–641.

[2] NEWTON, I. *De methodis serierum et Fluxionum.* Manuscript, 1671.

[3] TURING, A. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc. 2*, 42 (1936), 230–265.

`tm-alpha`

The achievements in [New71, Ein05, Tur36] have changed the world.

[Ein05] Albert Einstein. Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik*, 323(113):639–641, March 1905.

[New71] I. Newton. *De methodis serierum et Fluxionum.* Manuscript, 1671.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.

`tm-elsart-num`

The achievements in [1, 2, 3] have changed the world.

[1] I. Newton, De methodis serierum et Fluxionum, Manuscript, 1671.

[2] A. Einstein, Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig?, Annalen der Physik 323 (113) (1905) 639–641.

[3] A. Turing, On computable numbers, with an application to the Entscheidungsproblem, Proc. London Maths. Soc. 2 (42) (1936) 230–265.

`tm-ieeetr`

The achievements in [1, 2, 3] have changed the world.

[1] I. Newton, *De methodis serierum et Fluxionum*. Manuscript, 1671.

[2] A. Einstein, ``Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig?,'' *Annalen der Physik*, vol. 323, no. 113, pp. 639–641, March 1905.

[3] A. Turing, ``On computable numbers, with an application to the Entscheidungsproblem,'' *Proc. London Maths. Soc.*, vol. 2, no. 42, pp. 230–265, 1936.

`tm-plain`

The achievements in [2, 1, 3] have changed the world.

[1] Albert Einstein. Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik*, 323(113):639–641, March 1905.

[2] I. Newton. *De methodis serierum et Fluxionum.* Manuscript, 1671.

[3] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.

`tm-siam`

The achievements in [2, 1, 3] have changed the world.

[1] A. EINSTEIN, *Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig?*, Annalen der Physik, 323 (1905), pp. 639–641.

[2] I. NEWTON, *De methodis serierum et Fluxionum*, Manuscript, 1671.

[3] A. TURING, *On computable numbers, with an application to the Entscheidungsproblem*, Proc. London Maths. Soc., 2 (1936), pp. 230–265.

`tm-unsrt`

The achievements in [1, 2, 3] have changed the world. Yet, would they be qualified as excellent^{+++} according to modern criteria for research proposals?

[1] I. Newton. *De methodis serierum et Fluxionum*. Manuscript, 1671.

[2] Albert Einstein. Ist die Trägheit eines Körpers von seinem Energieinhalt abhängig? *Annalen der Physik*, 323(113):639–641, March 1905.

[3] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Maths. Soc.*, 2(42):230–265, 1936.

6.5.4 Quick survey of BIBTEX

BIBTEX files are raw text files that can be edited using any text editor. When editing them using TEXMACS, a few special rules apply that will be discussed in section 6.5.5 below. But let us first survey the most important facts about the BIBTEX format. For more details and special features of lesser interest for ordinary users, we refer to [38, appendix B] and [16, chapter 13].

In fact, BIBTEX is really a program that takes a BIBTEX file and a list of citations as input and produces the corresponding bibliography as output. Whenever you edit BIBTEX files by hand, it is important to follow the BIBTEX syntax. Otherwise, errors may occur when running BIBTEX and the resulting bibliography might be badly formatted. Recall that TEXMACS contains its own replacement for the BIBTEX program; to use this native replacement, your bibliography style must be prefixed by `tm-`; otherwise, TEXMACS will run BIBTEX whenever it needs to generate a bibliography.

A BIBTEX file consists of a list of BIBTEX entries, which are usually separated by empty lines. Each entry consists of three main parts: a type specifier, an identifying key, and a comma-separated list of data fields. The file should not contain any "duplicate" entries with the same identifying key; indeed, any references to such entries would be ambiguous. Each data field of an entry consists of a field type and a value:

```
@Entry_type{key,                       @Book{Bor28,
  field_type_1 = field_value_1,          author =    {\'E. Borel},
  field_type_2 = field_value_2,          title =     {Le\c{c}ons sur les
  ...                                                 s\'eries divergentes},
  field_type_n = field_value_n,          publisher = {Gauthier-Villars},
}                                        year =      {1928}
                                       }
```

The field values should be enclosed between curly brackets `{val}` or quotes `"val"`. The values themselves are in LaTeX format, but a few additional rules apply that will be discussed below. Notice that entry and field types are not case-sensitive, contrary to the key and the field values.

BibTeX supports a fixed set of entry types that are listed below. For each entry type, there are three kinds of fields: required fields, optional fields and ignored fields. For instance, the entry type `Article` requires you to specify the `author`, `title`, `journal`, and `year` (otherwise, an error will occur and the resulting bibliography will probably be badly formatted). In addition, you may specify the optional fields `volume`, `number`, `pages`, `month`, and `note`. Such optional fields will also appear in the generated bibliography using an appropriate layout. All other field types are ignored for the `Article` entry type, even though they may contain interesting information such as an abstract, a hyperlink to the paper, etc. BibTeX entries that are downloaded from the web generally contain several ignored fields that may be useful for bookkeeping or other purposes.

6.5.4.1 Entry types

BibTeX supports the following standard entry types:

Article. An article in a journal or magazine. Required fields: `author`, `title`, `journal`, `year`, `volume`. Optional fields: `number`, `pages`, `month`, `note`, `key`.

Book. A book with an official publisher. Required fields: `author` or `editor`, `title`, `publisher`, `year`. Optional fields: `volume/number`, `series`, `address`, `edition`, `month`, `note`, `key`.

Booklet. A work that is printed and bound, but without any explicit publisher or sponsoring institution. Required field: `title`. Optional fields: `author`, `howpublished`, `address`, `month`, `year`, `note`, `key`.

Conference. The same as `Inproceedings`.

Inbook. An untitled part of a book, such as a chapter, a section, or a range of pages. Required fields: `author/editor`, `title`, `chapter/pages`, `publisher`, `year`. Optional fields: `volume/number`, `series`, `type`, `address`, `edition`, `month`, `note`, `key`.

Incollection. A part of a book with its own title. Required fields: author, title, booktitle, publisher, year. Optional fields: editor, volume/number, series, type, chapter, pages, address, edition, month, note, key.

Inproceedings. An article in the proceedings of a conference. Required fields: author, title, booktitle, year. Optional fields: editor, volume/number, series, pages, address, month, organization, publisher, note, key.

Manual. Technical documentation such as a user manual for a program. Required field: title. Optional fields: author, organization, address, edition, month, year, note, key.

Mastersthesis. A Master's thesis. Required fields: author, title, school, year. Optional fields: type, address, month, note, key.

Misc. To be used when nothing else fits. No required fields. Optional fields: author, title, howpublished, month, year, note, key.

Phdthesis. A Ph.D. thesis. Required fields: author, title, school, year. Optional fields: type, address, month, note, key.

Proceedings. The proceedings of a conference. Required fields: title, year. Optional fields: editor, volume/number, series, address, month, publisher, organization, note, key.

Techreport. A report published by some institution, usually numbered within a series. Required fields: author, title, institution, year. Optional fields: type, number, address, month, note, key.

Unpublished. A document having an author and title, but not formally published. Required fields: author, title, note. Optional fields: month, year, key.

Unfortunately, BIBTEX has no special entry type for preprints. You may use Unpublished, Techreport or Misc for this purpose, with the appropriate information on how the preprint was published in the optional fields. In the future, TEXMACS might support a few additional entry types for preprints and modern kinds of electronic publications.

6.5.4.2 Field types

The required and optional field types for the standard BIBTEX entry types are listed below:

address. Publisher's address, usually just the city, but possibly the full address.

annote. An annotation for some less common annotated bibliography styles.

author. The name(s) of the author(s).

booktitle. Title of the book, if only part of it is being cited.

chapter. Number of the chapter.

crossref. BIBTEX key of the cross-referenced entry.

edition. Edition of a book, in long form, such as "First" or "Second".

editor. The name(s) of the editor(s).

howpublished. How the work was published.

institution. The institution that was involved in the publishing, but not necessarily the publisher.

journal. The journal or magazine the work was published in.

key. A hidden field which specifies how to sort the entry in a bibliography. Notice that this field is different from the entry's BIBTEX key that is used for citations and cross-references.

month. The month of publication or creation.

note. Miscellaneous extra information such as the URL where to download a preprint.

number. The "(issue) number" of a journal, magazine, or tech-report, if applicable.

organization. The sponsor of a conference.

pages. Page numbers, separated either by commas or dashes.

publisher. The publisher's name.

school. The university or school where the thesis was written.

series. The series of books the book was published in, such as *Astérisque* or *Lecture Notes in Computer Science*.

title. The title of the work.

type. The field overriding the default type of publication (e.g. "Preprint" or "Research Note" for `Techreport`, "Habilitation" for `Phdthesis`, or "Section" for `Inbook/Incollection`).

volume. The volume of a journal or multi-volume book.

year. The year of publication or creation.

BIBTEX entries that were downloaded from the web often include other non-standard fields such as a DOI or publisher-specific information. These fields are ignored by the traditional BIBTEX styles, but can be useful for other bibliographic software or non-standard styles. In the future, TEXMACS might provide its own non-standard extensions of this kind.

6.5.4.3 Field values

Field values should be enclosed between curly brackets or quotes, as in

```
journal = {Annalen der Physik}    or
journal = "Memoirs of the AMS"
```

Numbers do not require any brackets or quotes:

```
volume = 1234
```

We recall that field values are specified in LATEX format. In particular, accented characters are entered as in LATEX:

```
author = { \'Emile Borel }
```

BIBTEX also follows a few special rules concerning names and titles.

Names. BIBTEX expects a name or a list of names for the values of `author` and `editor` fields. You must use and as a separator when entering more than one name:

```
author = {G. H. Hardy and E. M. Wright}
```

BIBTEX names consist of four parts: a first name, an optional "von" part, a last name, and an optional Jr. part. Simple names without von-parts or suffixes can be entered in two equivalent ways:

```
author = {Albert Einstein}    or
author = {Einstein, Albert}
```

You are allowed to use abbreviations instead of first names:

```
author = {C. St. J. A. Nash-Williams}    or
author = {Nash-Williams, C. St. J. A.}
```

BIBTEX recognizes "von" parts via the use of lowercase between an uppercase first and last names:

```
author = {John von Neumann}
```

Unfortunately, the way these "von"-parts are handled by most BIBTEX styles is far from satisfactory. For instance, in order to correctly sort the author of this book in a bibliography, you must use

```
author = {Hoeven, Joris van der}
```

instead of

```
author = {Joris van der Hoeven}
```

Special suffixes such as Jr. or IV in "Henry IV" are entered as follows:

```
author = {Johnson, Jr., John}
```

Titles. BIBTEX styles may change the case of titles when appropriate. Titles of books are usually capitalized, unlike those of articles. In BIBTEX, it is recommended to capitalize titles in the same way as in the original work. For instance:

```
title = {Factoring Polynomials with
         Rational Coefficients}
```

Inside an actual bibliography, this title is typically transformed as follows:

[LLL82] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

In order to prevent names of people or places to appear in lower case, you may surround them by curly braces:

```
title = {An Algorithm for the Machine Calculation
         of Complex {Fourier} Series}
```

Inside a bibliography, such a title might yield

[CT65] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

The same mechanism applies to German nouns, which must always be capitalized.

6.5.5 Editing BIBTEX entries with TEXMACS

TEXMACS comes with an integrated editor for BIBTEX entries. This editor is used by default when opening a BIBTEX file with the `.bib` extension. It is also used for managing your personal bibliographic database. After enabling the database tool using Tools ▸ Database tool, you may open your personal bibliographic database via Data ▸ Open bibliography. In that case, and as detailed in section 6.5.6 below, the editor also provides tools for querying and modifying the database.

Assume that we have opened a BIBTEX file or a bibliographic database. Then new items can easily be added using Insert ▸ Database entry. When creating a new entry, required fields appear in dark blue, alternative fields in dark green and optional fields in light blue. The special field inside the header of your entry contains the identifying BIBTEX key that is used for citations. When editing a field, you may use ↩ to confirm it and jump to the next one (empty optional fields will automatically be removed when doing this). When the cursor is inside a bibliographic entry, additional fields may also be inserted using Focus ▸ Insert above and Focus ▸ Insert below.

When editing BIBTEX entries in plain text format, we saw in section 6.5.4.3 that various contortions were necessary for editing names and titles. Formulas also need to be "encoded" using LATEX pseudo-code. These disadvantages disappear when using the integrated editor, although a few new rules need to be followed in order to get your entries right:

- When entering names (inside "`author`" or "`editor`" fields), use the name tag for specifying last names (with Insert ▸ Last name or ⇧F6). For example, "Albert Einstein" should be entered as "Albert EINSTEIN" or as "A. EINSTEIN". Special particles such as "von" can be entered using Insert ▸ Particle. Title suffixes such as "Jr." can be entered similarly using Insert ▸ Title suffix. Finally, the , key is mapped to the name separator "`and`".

- When entering titles, do not capitalize, except for the first character and names or concepts that always must be. For example, use "Riemannian geometry" instead of "Riemannian Geometry" and "Differential Galois theory" instead of "Differential Galois Theory".

In addition, when pressing ↩ in an `author` or `editor` field for which you forgot to "tag" the last name(s), TEXMACS automatically attempts to determine all last names and insert the missing name tags. For instance, "G. H. Hardy and E. M. Wright" is automatically transformed into "G. H. HARDY and E. M. WRIGHT". You can bypass this correction step by pressing ⇧↩ instead of ↩.

While editing a bibliography, it is useful to keep in mind that external BIBTEX entries can easily be imported: it suffices to copy them using the mouse and then paste them into the editor using Edit ▸ Paste from ▸ BibTeX. Similarly, you may copy BIBTEX entries inside the editor using Edit ▸ Copy to ▸ BibTeX and then paste them into other programs.

When using TEXMACS for editing existing BIBTEX files, it is also useful to know that TEXMACS attempts to minimize the number of changes that are made when re-exporting or re-importing a file after some modifications. For example, only modified entries will be altered on disk when saving a modified file. This default behavior can be overridden via Edit ▸ Preferences ▸ Convert ▸ BibTeX.

6.5.6 Managing your personal bibliographic database

TEXMACS contains an integrated database tool that allows you to maintain a shared bibliographic database for all your documents. The tool offers various advantages:

- It is easy to search for specific information in the database and customize how matching items should be sorted and displayed.

- When a colleague sends you a file with embedded data, it is possible to automatically import the data into your personal database.

- The database tool maintains the full history of all data, while keeping track of the persons behind changes. This is useful for the resolution of conflicts when importing data from a file sent by a colleague.

We recall that the database tool can be enabled using Tools ▸ Database tool.

6.5.6.1 Searching and rendering of entries in the database

Your personal bibliographic database can be opened using Data ▸ Open bibliography. The database usually contains many entries, so only a limited number of entries are shown. The precise entries that should be shown and the precise way in which they should be rendered can be specified on the *database toolbar* at the bottom of your window:

- The Search field allows you to search for particular entries. The search text should be a list of words separated by spaces. A database entry matches whenever each word occurs in the entry. Occurrences may be anywhere: in the name, in the title, in the name of the journal, or even in invisible fields such as the contributor of the database entry.

- The Order fields allows you specify the name of field on which the entries will be sorted. The little icon ▼ or ▲ next to this field indicate whether entries should be sorted in ascending or descending order.

- The Limit field specifies the number of matches that should be displayed. When using a large number such as 1000, beware that this may slow down the editor.

- The rightmost item on the database toolbar allows you to specify the rendering: Detailed is the default rendering used for editing the entries, Folded only shows the main information about the entries, and Pretty uses a typical rendering for print. When using a folded or pretty rendering you may switch to a detailed rendering by clicking on the entry; this will allow you to edit it.

6.5.6.2 Insertion and removal of entries

Visible database entries (depending on your search and display options) can directly be edited as described in section 6.4.2. New entries can be inserted using Data ▸ New entry and then edited in the same way. Note that modified or newly entered entries remain visible even if they do not correspond to your search criteria: the search filter is only applied when you explicitly modify one of the search or rendering options.

Once you have finished typing an entry, it can be inserted into your personal database using ⌥↩, ✓, or Data ▸ Confirm entry. If all fields of an entry have been filled out, then pressing ↩ in the last field will have the same effect. A selection of several entries can be simultaneously inserted using the same keystroke, ⌥↩. Saving the document using File ▸ Save amounts to confirming all currently visible entries.

Existing entries can be removed using ⌥⌫, , or Data ▸ Remove entry. This either applies to a selection of entries or to the current entry that contains your cursor. Note that "cutting" one or more entries in the editor will not have the same effect: this will just make them invisible, but not remove them from the database.

6.5.6.3 Importing and exporting entries

Entries from an external BibTeX file or from another database can be imported into your personal database. Assuming that you opened your personal bibliographic database using Data ▸ Open bibliography, external files can be imported using Data ▸ Import. If you are rather editing an external BibTeX file or another database, then you may incorporate entries into your personal bibliographic database using Data ▸ Import entries in buffer. It is also possible to import the entry that contains your cursor using Insert ▸ Import entry or . More generally, you can import a selection of entries using Data ▸ Import selected entries, Insert ▸ Import selected entries, or .

Conversely, the bibliographic entries in your personal database can be exported to a BibTeX file: assuming that you opened your personal bibliographic database using Data ▸ Open bibliography, this is done using Data ▸ Export. In order to export a selection of entries, you may use Data ▸ Export selected entries. We also remind you of the possibility to copy a selection to the clipboard in BibTeX format using Edit ▸ Copy to ▸ BibTeX.

Note that the precise location where your personal database is stored can be customized using Data ▸ Storage. Whenever convenient, this also allows you to switch between several databases.

6.5.6.4 Versioning

The database keeps track of the full history of all entries. As in the case of popular versioning tools such as SVN or GIT, the history includes information about the dates when modifications were made and by whom.

Every entry in the database also keeps a record of all previous versions of the entry. This is useful in the following scenario: you send an entry to a colleague, who modifies it, and sends it back. When importing the modified entry, the datatool automatically recognizes that it is a newer version of an existing entry. However, if you modified the entry as well in the meantime, then your personal modification will be preferred.

The above mechanism applies automatically when a colleague sends you file that contains a bibliography: TeXmacs will attempt to enrich your personal database with all the bibliographic entries that it can find in the file. If you do not like this feature and wish to retain more manual control, then you can disable it in Data ▸ Preferences. In that case, importations have to be done manually, as described in section 6.5.6.3.

6.5.7 Local bibliographies

Most of the time, it is a good idea to maintain a global database with up-to-date bibliographic entries, while possibly sharing this information with your colleagues. However, for the final version of a paper, it may happen that some twiddling is necessary that is specific to the paper. In that case, you may want to change some of the bibliographic entries, without integrating them into your personal database (or to the database of any of your colleagues). For this particular situation, TeXmacs makes it possible to use a local bibliographic database that is attached to your document.

Assuming that your document indeed contains a bibliography, the local database is opened using Document ▸ Bibliography. By default, this database is maintained automatically: it contains all bibliographic entries that were needed for the compilation of your bibliography and that were fetched from your personal database or a specified BibTeX file. Now you can edit the local bibliography in a similar way as a BibTeX file. If you save your modifications, then TeXmacs determines all entries that were changed; these entries will be marked as "local" and they cease to be maintained automatically. Your local changes will be taken into account when recompiling the bibliography, but they will not be imported automatically into your global database when reopening your document. Of course, it remains possible to manually import entries from a local bibliography into your global database using Insert ▸ Import entry or .

6.5.8 Multiple bibliographies

In a thick book or collection of papers, you may want to include a separate bibliography for each chapter or paper, together or not with one for the entire book. Multiple bibliographies can be generated in a similar way as multiple tables of contents or multiple indexes (see section 6.4.5): each bibliography is given its own identifier, the default one being `bib`. Using Insert ▸ Link ▸ Alternate ▸ Bibliography, you may next specify which bibliography to use when inserting citations or the bibliography itself.

6.6 Large documents

When writing really large documents such as books or PhD theses, there exist several good reasons for breaking up your work into smaller parts. First of all, the editor may become slow on large documents and editing an entire book in a single window may not be that convenient anyway. Secondly, if your book is the fruit of a collaboration, then it is important that all coauthors can work independently and concurrently on different chapters.

The recommended way to prepare a book with TeXmacs is to first create a master project file that will contain the entire book, say `mybook.tm`. You should use Tools ▸ Project ▸ Use as master in order to declare this file to be the master. The master typically consists of title information, automatically generated content (the table of contents, a bibliography, etc.), and links to the individual chapters.

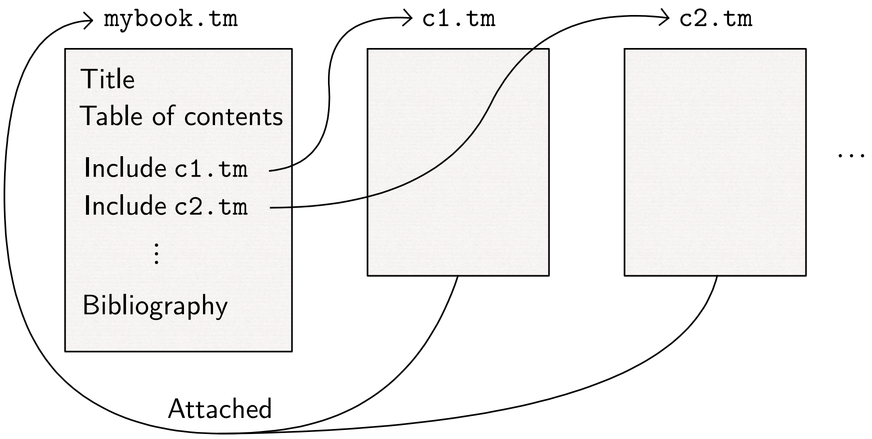


Figure 6.6. Schematic representation of a book project with a master and individual chapters.

You should also enable paper mode (Document ▸ Page ▸ Format ▸ Page rendering ▸ paper) in order to get the page numbers right when updating the document. Notice that the project file is rarely edited itself: this only happens when you modify the main structure of the book, e.g. by adding a new chapter. The main book style and other global layout properties should also be specified in the master file.

Now assume that we have a separate file `c2.tm` with a chapter that we wish to include in the book. Then you should first include the chapter at the appropriate place in the master file using Insert ▸ Link ▸ Include. In the chapter file itself, you also have to attach the chapter to its master, using Tools ▸ Project ▸ Attach master. If all this is done correctly, then the master file and the individual chapter files should be accessible through the top-level Project menu. Tables of contents, bibliographies, etc. can directly be inserted into the master file in the usual way. See Figure 6.6 for a summary.

Since the actual editing is done in the chapter files, it is most convenient to select papyrus mode (Document ▸ Page ▸ Format ▸ Page rendering ▸ papyrus) for them. For consistency, you may still wish to use the same book style and global layout properties as for the master file, although this is not mandatory.

Page numbers, chapter numbers, bibliographic citations, and cross references between different sections are taken care of by TeXmacs in a semi-automatic way: from time to time, you need to update the master file in order to get all numbers and further references right. It is recommended that you save the master file after such updates; this ensures that references remain up to date after closing the editor and re-opening one of the chapters. For the computation of page numbers, it is also important to enable the paper mode for the master file. Whenever you modify the main book structure, some individual chapters occasionally may need to be updated as well.

Chapter 7

Advanced layout facilities

Typographic style is a complex topic that we cannot hope to cover in a single section or chapter of this book. One of the bibles on the subject is Bringhurst's *The Elements of Typographic Style* [4]. Lawson's *Anatomy of a Typeface* [40] focusses more specifically on fonts.

It the first section, we describe a few general principles that may be useful to keep in mind when working on the layout of your document. In the remainder of the chapter, we continue with a description of some of the more advanced layout features inside TeXmacs that can be used to put these principles into practice.

7.1 General style considerations

Purpose

Whenever you are inclined to use a special style for some portion of text, rule number one is to ask yourself the question "*Why?*" The more conscious you are of the precise purpose of a style element, the more likely you will opt for a graphical presentation that correctly conveys its intent. Take for instance the graphical interface elements "menu" and "keyboard shortcut" in this book. Their purpose is to make it immediately clear for the reader that File ▸ Open corresponds to a TeXmacs menu entry and - - > to a series of keystrokes. In this case, we chose a presentation that mimics physical appearance: the font for the menus is the same as the menu font inside the editor and the little boxes around the keystrokes remind of the keys on your keyboard.

Part of these considerations can be aesthetic: alternate fonts are often chosen for their contrast with the main font. This is why we chose a sans serif font for the section titles in this book. For the main font, we preferred the more recreational TeX Gyre Pagella font (a variant of PALATINO) over the more well-known but sterner Computer Modern Roman and Stix fonts.

Uniformity

Another important rule is to stick to your own design decisions. For example, if you decide to use an italic font for *important text*, then you should do so throughout your document: nothing is more confusing for the reader than suddenly switching to a bold font for no apparent reason.

By its very design, TeXmacs makes it easy to achieve uniformity: the most common style elements are directly available through the interface and their rendering can be customized when needed. The macro system also allows to add new style elements of your own. Still, even though the interface invites you to use appropriate markup in a systematic way, you should not forget to do so!

Minimality

A design principle that is related to the two above ones is "minimality": do not use more style elements than needed and only use alternate fonts for well-identified purposes. Some advertisers love to "impress" people through the use of an overwhelming number of flashy fonts, but most of the time this is just a confusing distraction from the main message, even in the case of commercial adds (successful businesses such as APPLE and GOOGLE tend to favor a minimalistic design).

When introducing new style elements using the macro system, it is also recommended to introduce new macros only if you really need them. In a botanical guide, it would in principle be possible to introduce separate macros for the names of mosses, flowers, conifers, and cycads. But would this really be a good idea? Maybe so, if they require four different renderings; otherwise, a single macro "name" or "plant" should suffice.

Standard conventions

It is good practice to learn common typographic and linguistic conventions. Standard references on these topics are [4, 31, 5]. Notice that conventions depend on the language you are writing in. For example, the characters : ; ! ? are preceded by whitespace in French [17], but not in English.

It is recommended to always follow standard conventions unless you have a good reason to ignore them: bad spelling or typography may lead to misunderstandings or irritate your reader. Unfortunately, some conventions are quite arbitrary (the English metric system) or even defy common logical sense: in this "quote," the comma is inside the quote (following standard English conventions[7.1]), even though it is not part of the quote, strictly speaking. Don't hesitate to break flawed rules: standards might ultimately follow your lead [46].

7.1. The convention is due to the observation that "comma," *looks* a bit nicer than "comma", typographically speaking. As a compromise, we suggest to put the "comma", outside the quote, but to precede it by negative whitespace.

Specialization

General conventions may also need to be overruled in specific circumstances. For instance, in ordinary literary books, it is best to use a fixed vertical spacing between lines. In the case that a single line of a paragraph does not fit on a page (an orphan), one often moves an additional line to the next page. This results in a shorter page, but having such a shorter page is considered a lesser evil than a slight increase of the vertical space between all lines on the page (so as to keep a fixed page height). But now consider the case of a mathematical book: it is likely to contain many formulas of wildly varying heights. This makes it unrealistic to require a rigid interline space. In this particular case, it therefore makes sense to use a slightly stretchable interline space, and strive for a fixed page height.

Specific contexts may also call for the temporary suspension of general rules. For example, modern bibliographies often contain hyperlinks that are hard to break. It is therefore wise to somewhat relax the rules for line breaking inside bibliographies. A similar remark applies for the output formulas in computer algebra sessions.

Looking for the right feature

As soon as you identified a style element that should serve a certain purpose, you have to start thinking about the best way to render this element. If you are lucky, then TeXmacs already contains a markup element that does precisely what you want. Otherwise, you should search hard for the appropriate features in TeXmacs that allow you to achieve the desired effect in a systematic and automated manner. This chapter describes several of the finer tricks that you may find useful.

A similar remark applies to the global layout of your document: if you need a particular design, then it is a good practice to always search for an existing TeXmacs style that comes close to what you want. At a second stage, you may select additional style packages or customize some of the macros. If you decide to change major parameters such as the page size or margins, then it may be useful to be aware of proportions that are particularly pleasant to the eye. We refer to [4, chapters 2 and 8] for more information.

Trade-offs

There is often a trade-off between various design goals. For example, alternate fonts should typically be sufficiently similar to the main font (in order to "mix" well), but also sufficiently distinctive (in order to avoid confusions). A similar issue came up in the present book: it is natural to use specific markup for various purposes: important concepts, menus, keyboard shortcuts, source code, TeXmacs markup, names of TeXmacs styles or environment variables, etc.

However, one quickly runs out of the available fonts if one insists on having a different rendering for each of these style elements. When facing these kinds of dilemmas, it is often impossible to find a completely satisfactory solution, and one is condemned to opt for the lesser evil. Yet, the general rules that we described so far may act as a guide.

Automatic versus hand-crafted

Typography is an "art" that has been developed ever since the invention of printing. Although computer programs have become pretty good typographers, machines do not always manage to keep up with human competition.

The first reason for this is that the implementation of high quality typesetting algorithms requires a lot of effort. Besides the fact that only few developers are willing to make this investment, it is not always clear how to mimic traditional human skills. Although TeXmacs attempts to keep pace with state-of-the-art technology, a lot progress can and probably will be made in the future.

Secondly, some of the finer details are a matter of taste and may depend on the actual content or purpose of a document. The typesetting process in particular depends on many parameters. Although default values can be implemented that work reasonably well on the average, you may wish to override these settings for particular purposes (recall the above discussion about "specialization"). For this, it is important to understand the typographic challenge that such a parameter was meant to address.

Personal taste

In the end, your preferred layout of documents remains a highly subjective matter. The above considerations are meant as a first step to sharpen your taste and make you more conscious of the reasons why you prefer certain things. We have already mentioned a few reasons behind our own design choices in this book. Let us end with one more example.

You may have noticed that we chose to separate distinct paragraphs by vertical whitespace. In many books, the first lines of paragraphs are rather indented. Why did we prefer the first option? By its very nature, this book contains a wide variety of markup elements and figures. This causes a lot of "disorder" that would be further aggravated by indenting the first lines of paragraphs.

7.2 Line breaking

One of the most delicate tasks of the typesetter is to break paragraphs into successive lines. The line breaking problem is most difficult for narrow paragraphs that you can see in newspapers or magazines. Whenever a long word does not fit on a line, moving it to the next line typically requires the insertion of a lot of whitespace. If a paragraph is justified, then this leads to annoyingly large spaces between words. Professional typesetters use a combination of techniques to deal with this problem:

- Through the use of high quality line breaking algorithms.
- By hyphenating certain words.
- Through the manual specification of good or bad break points.
- By adjusting the spacing between individual characters using microtypography.

Let us now describe these techniques in more detail.

7.2.1 The first-fit versus the global method

The simplest method to break lines is the so-called "first fit" algorithm. It processes the lines one by one and simply finds the best possible break for each line. One important drawback of this algorithm is that a much better break can sometimes be obtained for a given line by using a non-optimal break on one of the previous lines.

By default, TEXMACS therefore uses a so-called "global" line breaking algorithm, that takes into account *all* possible ways to break a paragraph into successive lines, and then selects the solution that minimizes ugly breaks among all these possibilities. One minor disadvantage of this method is that the line-breaking of a paragraph may radically change while you are typing. In particular, typing new words at the end of a paragraph may alter the line breaks at the start. If you find this annoying, then you may wish to opt for the first-fit algorithm.

The algorithm that TEXMACS uses for line breaking can be specified in Document ▸ Paragraph ▸ Line breaking: the global method corresponds to professional, and the first fit strategy to normal.

7.2.2 Hyphenation

As soon as the typesetter is unable to find suitable breaks between words, it will attempt to hyphenate the words themselves. The permitted hyphenations strongly differ from one language to another. For this reason, it is important to always specify the correct language for your documents in Document ▸ Language. If you need to combine text written in more than one language, then you may locally change the language using Insert ▸ Language.

It is hard to fully automate hyphenation. Some English words such as "record" may even admit different hyphenations depending on their meaning: please re-cord this world rec-ord. In science, it is also common to invent new words such as "transserial" or "exoplanetary", for which standard hyphenation patterns might lead to incorrect results. It is always possible to manually override the way a word should be hyphenated: select the word using the mouse, click on Format ▸ Hyphenate as, and, at the prompt, insert "-" characters at those places where you wish to allow for hyphenations.

Ordinary words are hyphenated using "-". Inside verbatim text, words may also be hyphenated after slashes; this is useful for long hyperlinks in the references. Long numbers are hyphenated using backslashes, as in the following example:

```
Pari] 200!

%1 = 7886578673647905035523632139321850622951359776871732632947425\
3324435944996340334292030428401198462390417721213891963883025764279\
0242637105061926624952829931113462857270763317237396988943922445621\
4516642402540332918641312274282948532775242424075739032403212574055\
7956866022603190417032406235170085879617892222278962370389737472000\
0000000000000000000000000000000000000000000000
```

7.2.3 Manual line breaking

If, for some reason, you don't like a particular line break, then you may manually override it. This can be done in two ways: by specifying suitable breaks or by forbidding bad breaks.

Suitable breaks can be forced explicitly using Format ▸ Break ▸ Line break. If paragraphs are justified, then explicit breaks will appear at the right-hand border. Alternatively, you may insert "new lines" using Format ▸ Break ▸ New line. This causes the typesetter to start a new line without stretching the current line to its right border: see Figure 7.1.

You may explicitly forbid a break using Format ▸ Break ▸ No line break. Such prohibitions are most frequent around spaces. For this reason, $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ also provides so-called "non-breaking spaces", which can be inserted using ␣ ⇥. For instance, it is recommended to systematically precede theorem and equation numbers by non-breaking spaces, as in Theorem 123 and Equation (1.2.3). Besides non-breaking spaces, it is also possible to insert "non-breaking hyphens", using - ⇥.

$\text{T}_\text{E}\text{X}_\text{MACS}$ also provides a way to forbid line breaking altogether inside a given region of text: just select the "unbreakable text" and apply Format ▸ Break ▸ Group ▸ Horizontal. This mechanism may in particular serve to disable line breaking inside subexpressions of mathematical formulas.

One disadvantage of explicit line breaks is that they tend to provoke disaster whenever you make further changes in a paragraph. Indeed, a perfect explicit break near the end of a line may suddenly find itself at the start of the next line, thereby causing it to be filled with whitespace. By contrast, explicit prohibitions at inappropriate places will simply be ignored. Unless you are preparing the very final version of a document, explicit line breaks should therefore be avoided.

Here comes a break inserted using Format ▸ Break ▸ Line break.	Here comes a break inserted using Format ▸ Break ▸ New line.

Figure 7.1. Difference between a "line break" and a "new line".

7.2.4 Microtypography

The term *microtypography* designates a range of methods for improving the readability and appearance of text [59]. The prefix "micro" stems from the fact that we allow ourselves to alter the spacing and even shapes of individual characters to reach this goal. One of the chief goals of microtypography is to improve the justification when paragraphs need to be broken into several lines.

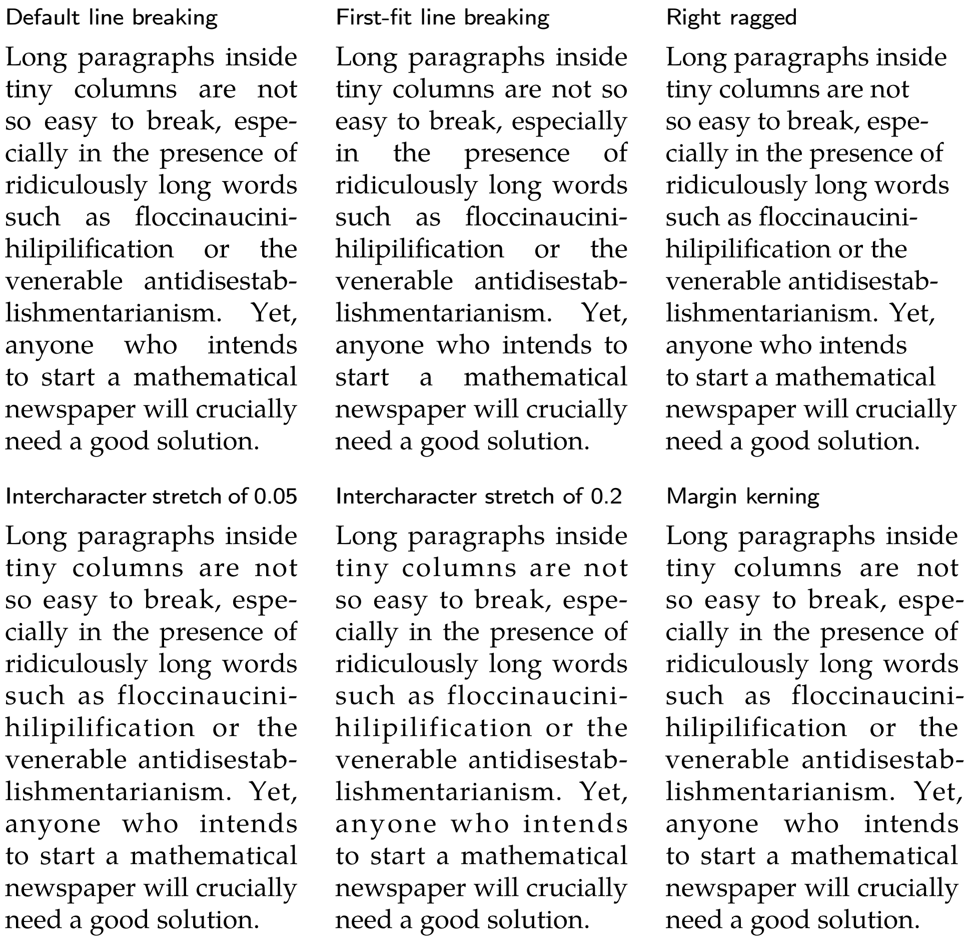
One microtypographic trick that is used by $\text{T}_\text{E}\text{X}_\text{MACS}$ is called *tracking*. The idea is to slightly increase the spacing between individual characters in order to reduce the amount of whitespace between words. The maximal stretching factor can be specified via Format ▸ Paragraph ▸ Advanced ▸ Intercharacter stretching. A value of `0.05` allows for a maximal increase of the width of each character by 5%; larger values are generally not recommended. Tracking is not only supported for individual words, but also for subexpressions inside mathematical formulas, as in the following computer algebra output:

```
(%i1) expand ((1/2! * a + 1/6! * b + 1/8! * c)^5)
```

(%o1)
$$\frac{c^5}{106562062388507443200000}+\frac{b\,c^4}{380578794244669440000}+$$
$$\frac{a\,c^4}{1057163317346304000}+\frac{b^2c^3}{339802494861312000 0}+\frac{a\,b\,c^3}{4719479095296000}+$$
$$\frac{a^2c^3}{26219328307200}+\frac{b^3c^2}{60679016939520000}+\frac{a\,b^2c^2}{561842749440 00}+\frac{a^2b\,c^2}{156067430400}+$$
$$\frac{a^3c^2}{1300561920}+\frac{b^4c}{2167107747840000}+\frac{a\,b^3c}{1504935936000}+\frac{a^2b^2c}{2786918400}+$$

$$\frac{a^3 b c}{11612160} + \frac{a^4 c}{129024} + \frac{b^5}{193491763200000} + \frac{a b^4}{107495424000} + \frac{a^2 b^3}{149299200} + \frac{a^3 b^2}{414720} + \frac{a^4 b}{2304} + \frac{a^5}{32}$$

Another technique is to stretch individual glyphs. On the one hand, *glyph expansion* allows for a further reduction of the amount of whitespace between individual words by increasing the widths of the glyphs. On the other hand, *glyph contraction* may also allow you to fit an extra word on a line via a decrease of the widths. Traditional typographers even did better by using special extended or condensed fonts for these purposes; such fonts are more elegant in combination with the main font than machine generated imitations through mathematical transformations.



Default line breaking

Long paragraphs inside
tiny columns are not
so easy to break, espe-
cially in the presence of
ridiculously long words
such as floccinaucini-
hilipilification or the
venerable antidisestab-
lishmentarianism. Yet,
anyone who intends
to start a mathematical
newspaper will crucially
need a good solution.

First-fit line breaking

Long paragraphs inside
tiny columns are not so
easy to break, especially
in the presence of
ridiculously long words
such as floccinaucini-
hilipilification or the
venerable antidisestab-
lishmentarianism. Yet,
anyone who intends to
start a mathematical
newspaper will crucially
need a good solution.

Right ragged

Long paragraphs inside
tiny columns are not
so easy to break, espe-
cially in the presence of
ridiculously long words
such as floccinaucini-
hilipilification or the
venerable antidisestab-
lishmentarianism. Yet,
anyone who intends
to start a mathematical
newspaper will crucially
need a good solution.

Intercharacter stretch of 0.05

Long paragraphs inside
tiny columns are not
so easy to break, espe-
cially in the presence of
ridiculously long words
such as floccinaucini-
hilipilification or the
venerable antidisestab-
lishmentarianism. Yet,
anyone who intends
to start a mathematical
newspaper will crucially
need a good solution.

Intercharacter stretch of 0.2

Long paragraphs inside
tiny columns are not
so easy to break, espe-
cially in the presence of
ridiculously long words
such as floccinaucini-
hilipilification or the
venerable antidisestab-
lishmentarianism. Yet,
anyone who intends
to start a mathematical
newspaper will crucially
need a good solution.

Margin kerning

Long paragraphs inside
tiny columns are not
so easy to break, espe-
cially in the presence of
ridiculously long words
such as floccinaucini-
hilipilification or the
venerable antidisestab-
lishmentarianism. Yet,
anyone who intends
to start a mathematical
newspaper will crucially
need a good solution.

Figure 7.2. Examples of various line breaking styles.

The glyph expansion and contraction factors can be controlled using Format ▸ Paragraph ▸ Advanced ▸ Character expansion and Character contraction. Manual horizontal magnification of fonts is also possible via Format ▸ Font effects ▸ Magnify horizontally. It is recommended to use a magnification factor between `0.95` and `1.05` if you wish to simulate glyph expansion. $\text{T}_\text{E}\text{X}_{\text{MACS}}$ finally implements the emulation of extended and condensed fonts in a way that preserves the stroke width: Format ▸ Font effects ▸ Extended and Condensed.

Besides the above strategies for avoiding large whitespaces, $\text{T}_\text{E}\text{X}_{\text{MACS}}$ also supports *protrusion*, which is equally known under the names *margin kerning* or *hanging punctuation*. This mechanism aims at aligning the left and right margins of the text in an optical manner, rather than rigidly fitting all text between two imaginary lines. Protrusion was first used in the very Gutenberg bible. You may enable it via Format ▸ Paragraph ▸ Advanced ▸ Use margin kerning.

7.3 Page breaking

After cutting your text into pieces that fit on successive lines, the typesetter moves on to page breaking and the assembly of the final pages of your document. Page breaking is similar to line breaking, with similar bells and whistles for controlling the process.

Again, $\text{T}_\text{E}\text{X}_{\text{MACS}}$ implements both a first-fit and a global algorithm for page breaking, which can be selected via Document ▸ Page ▸ Breaking ▸ Page breaking algorithm ▸ sloppy and professional. The global algorithm is used by default, but there are two main reasons why you may sometimes prefer the lower quality algorithm. First of all, when preparing the final version of a document, it is convenient to perform one last pass in order to correct suboptimal page breaks. The problem with global page breaking is that any corrections on later pages may cause earlier pages to be broken differently. Consequently, multiple passes may be necessary before all breaks will be to your satisfaction.

We also note that the global page breaking algorithm is more complex and therefore more time consuming. The algorithm may become particularly slow in the case of long documents that use a multiple column layout. When editing such documents in "paper mode", this causes the editor to become less responsive. With single column layouts, a similar slowdown only occurs for very long documents.

There are a few global style variables that influence precisely how page breaking is done. In Document ▸ Page ▸ Breaking ▸ Allowed page height reduction and Allowed page height extension, you may specify the amounts of space by which the default page height may be reduced or extended if no suitable break can be found. "Vertical space stretchability" specifies a multiplier for stretchable spaces. We recall that several vertical spaces are stretchable, in the sense that they may be increased if a page isn't full enough, or decreased when it is too full. For example, the whitespace around section titles, displayed formulas, and theorems is very elastic. The interparagraph and interline spaces are also stretchable, but to a lesser extent. When specifying a multiplier different from one for the Vertical space stretchability, you may further increase or decrease the elasticity of stretchable spaces.

As in the case of line breaking, you may force or forbid page breaks at specific points in the document. Such directives are indicated through markers. You may also specify whether you wish the directive to take effect before or after the line where the marker occurs. For instance, when doing Format ▸ Break ▸ Page break after ▸ New page, a marker is inserted in the text and a new page is started right after the line with the marker. Similarly, Format ▸ Break ▸ Page break before ▸ New double page starts a new page just before the line with the marker, and inserts one extra page if necessary in order to make sure that the new page appears at the right-hand side. This feature is useful when starting a new chapter in a book.

We recall that it is usually better to forbid bad page breaks rather than explicitly enforcing good ones. In order to prevent one specific bad break, you may use Format ▸ Break ▸ Page break before ▸ No page break or Page break after ▸ No page break. You may also forbid any page breaks to occur in a specified region of text using Format ▸ Break ▸ Group ▸ Vertical.

7.4 Multiple text flows

Simple documents contain a single *text flow* that is formatted into lines and pages by the typesetter. More complex documents often include footnotes, marginal notes, and floating figures or tables, that are interleaved with the main text in non-linear ways. In particular, each note and each floating object comes with its own text flow that is typeset independently, before being combined with the main text.

Note that footnotes and floating objects are mainly useful for documents that are intended to be readable on paper. In order to see where your notes will appear in the printed document, you must select a page rendering mode that shows the individual pages (using Document ▸ Page ▸ Page rendering ▸ paper; see section 3.10).

In Figure 7.3, we have shown an example of two pages in a book that contain two footnotes, a marginal note, a floating figure, as well as a floating table. Each of these objects is “anchored” in the main text flow. In the case of a footnote, the anchor is displayed in the form of a numeric superscript or a symbol such as †, ‡, etc. The anchors of marginal notes and floating objects are invisible on paper and indicated through “informative flags” on the screen. The

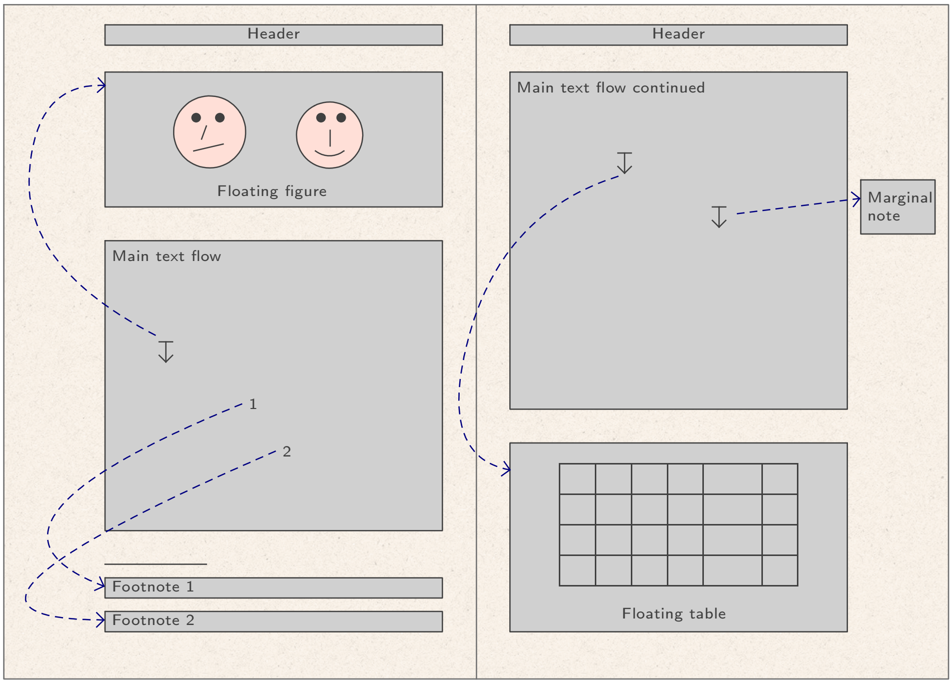


Figure 7.3. Two pages with notes and floats in a book.

anchors indicate on which page and at which position the notes or floating objects should be displayed.

The amount of whitespace between the main text flow and headers, footnotes, floating objects, etc. is controlled by style parameters. For instance, if the cursor is inside a marginal note, then the Focus ▸ Preferences menu (or, equivalently, the 🔧 menu on the focus toolbar) allows you to specify its width and the distance between the main text and the note.

7.4.1 Footnotes and marginal notes

Footnotes are started using Insert ▸ Note ▸ Footnote. They are numbered and you may jump between the main text and the footnote by clicking on their numbers or on the ⚓ icon. The style preferences in the Focus ▸ Preferences menu allow you to control the length of the separating footnote bar (Page fnote barlen), the amount of whitespace that separates footnotes from the main text (Page fnote sep), as well as the space between successive footnotes (Footnote sep). Notice that page breaking inside footnotes has not been implemented yet, whence footnotes are assumed to fit on a single page.

Warning. Basic footnotes only work inside the ordinary flow of text. If you want to put a footnote inside a table or other special markup, then you should use Insert ▸ Link ▸ Text for note and Reference to note. These menu items allow you to place the footnote and the corresponding reference to the footnote anywhere in the text, as shown in the example on the right.

First cell[7.2]	Second cell
Third cell	Last cell[7.3]

7.2. Text for the first footnote.

7.3. Text for the second footnote.

I am the living example of a marginal note.

Marginal notes are inserted using Insert ▸ Note ▸ Marginal note. They are similar to footnotes, except that they appear more prominently in the margins beside the main text, although their width is very limited. By default, when editing a document, $\mathrm{T_EX_{MACS}}$ reduces its margins on the screen with respect to the paper version. This is convenient if you want to make full use of your screen, but a nuisance in the presence of marginal notes. If you are using this feature, then it is recommended to set Same screen margins as on paper in Document ▸ Page ▸ Margins. The style parameters of marginal notes specify their width (Focus ▸ Preferences ▸ Marginal note width) and the separation space with respect to the annotated text (Marginal note sep).

The precise position of a marginal note can be controlled via the focus menu or toolbar. Its vertical alignment with respect to the anchor can be either Top, Center, or Bottom. Concerning the horizontal alignment, one may force placements in the Left or Right margin, or make this decision depend on the parity of the page number. If the left and right margins are different on left and right pages (as in Figure 7.3), then $\mathrm{T_EX_{MACS}}$ places marginal notes in the largest margin by default.

7.4.2 Floating figures and tables

Large objects such as figures and tables are often "indivisible" in the sense that they cannot be cut into several pieces by the page breaker. This causes a problem whenever the natural end of a page occurs in the middle of such a figure or table: the object does not fit on the page, but moving it to the next page will leave an unacceptably large amount of whitespace. One remedy is to manually move all large objects in a document to places that lead to appropriate page breaks. However, any subsequent modifications in the document may ruin this effort. A better solution is to let the typesetter take care of the problem by tagging the figure or table to be a *floating object*.

You may insert a floating figure, table, or algorithm using Insert ▸ Note ▸ Floating figure, Floating table, resp. Insert ▸ Note ▸ Floating algorithm. More generally, you may use Insert ▸ Note ▸ Floating object in order to create a floating object with arbitrary content. Existing figures, tables, and algorithms can also be turned into floating objects *a posteriori* by clicking on the icon or Focus ▸ Make floating.

Whenever you allow an object to float, the page breaker may freely move it to the top or bottom of the page, and even to the next page. Nevertheless, floating objects always appear in the same order as their anchors in the main text flow. The precise positions to which floating is allowed can be specified using Focus ▸ Allowed positions or . The style parameter that controls the amount of white-space between floating objects and the main text can be changed using Focus ▸ Preferences ▸ Page float sep.

When you are editing a floating object, you may jump to the corresponding anchor by clicking on the ⚓ icon or Focus ▸ Go to anchor. Conversely, if your cursor is just after the anchor, then clicking on one of these buttons will position your cursor at the start of the floating object. Note that page breaking inside floating objects has not been implemented yet. This means that floating objects should always fit on a single page.

7.4.3 Multiple columns

Publishers often use multiple column styles as an elegant way to reduce the paper footprint of publications. TEXMACS allows you to specify the number of columns both for an entire document or for a selected region of text, respectively using Document ▸ Paragraph ▸ Number of columns and Format ▸ Paragraph ▸ Number of columns. Two column layouts are the default for several standard document styles such as `acmconf` and `ifac`. The separation space between columns can be specified using Document ▸ Paragraph ▸ Column separation.

The typesetter attempts to balance the heights of adjacent columns as much as possible. For long columns from the top to the bottom of a page, this is done much in the same way as ordinary page breaking. The effect of height balancing is most visible on the last page or when varying the number of columns on the same page.

In multiple column formats, footnotes appear at the bottom of the column (instead of at the bottom of the page). Similarly, the typesetter prefers to move floating objects to the top or to the bottom of the same column.

On certain occasions, you may wish to select a different number of columns for the footnote or for the floating object. TEXMACS only knows how to do that for single column footnotes and floating objects, by selecting the anchor, opening Format ▸ Paragraph, and setting Number of columns ▸ 1. Wide footnotes and floats that are created in this manner always appear at the top or at the bottom of the page.

7.5 Finishing touches

During the preparation of the final version of a document, you may wish to address some of the more minor typesetting details. We already explained quite extensively how to control the line and page breaking mechanisms. It is also possible to adjust the horizontal and vertical spacing, or to tweak the extents of some formulas.

Most finishing touches are of a very visual nature; they typically correct a few remaining problems that keep hurting your eye. As such, the adjustments may crucially depend on particular line or page breaks, or the font that you use. This also means that any further change in the document has the potential of wrecking your work. For instance, assume that you inserted negative spaces into a formula so as to make it fit on a line. If a subsequent change in the paragraph results in removing a word from this same line, then it could become "underfull" instead of "overfull", and your negative spaces will do more harm than good.

For this reason, it is recommended to perform the final fine-tuning only at the very, very end. It is also worth it to invest a little thinking in how to make the adjustments. For example, if some specific page break does not suit you, then you have two options: preventing this break, or forcing a break at a more appropriate place. The second option may cause disaster if you insert new material on the previous pages: the forced break might very well migrate to the top of a page. Preventing ugly page breaks is the more robust solution: a bad break is likely to remain bad even after modifications earlier in the document.

Another example concerns fine-tuning the position of a subscript in a formula such as Γ_a. Indeed, this formula looks better if the a is moved slightly to the left, yielding $\Gamma_{\!a}$. If you use an absolute length such as `1mm` for this kind of repositioning, then the formula will not look as good whenever you switch to a different font size, or copy and paste it inside another subscript or superscript: $\Gamma_a + e^{\Gamma_a} + e^{e^{\Gamma_a}}$. This problem can be avoided by using a length unit that is proportional to the font size, such as `em`: $\Gamma_{\!a} + e^{\Gamma_{\!a}} + e^{e^{\Gamma_{\!a}}}$.

Due to evolutions inside TEXMACS, we notice that this kind of fine-tuning may occasionally lead to incorrect results when re-opening an adjusted document with a newer version of the editor. For example, the current version of TEXMACS produces Γ_a on our original example formula and $\Gamma_{\!a}$ when applying exactly the same adjustments as above. This is one side-effect of the maturing process of the typesetter, which has recently been educated to automatically move subscripts of Γ somewhat to the left. The precise correction depends on the font and can still be overridden via manual fine-tuning.

7.5.1 Horizontal spacing

In the Format ▸ Whitespace menu, you may find the available types of whitespace. For example, a horizontal whitespace of one centimeter can be inserted via Format ▸ Whitespace ▸ Horizontal space ▸ Stretchable, by entering `1cm` at the prompt. The length unit `cm` corresponds to a fixed space, but TEXMACS also provides so-called *stretchable* length units. For instance, the horizontal width of a space character between two words is `1spc`. The actual width of such a space is determined during line breaking: it may be somewhat reduced if we need to fit a lot of words on a line, or somewhat increased in order to fill a relatively empty line. General stretchable spaces therefore consist of a default, a minimal, and a maximal amount of space.

Keystroke	Space	Keystroke	Space
␣	1 spc	\ ! ↩	-0.2 spc
␣ ␣	1 em	\ , ↩	0.2 spc
␣ ␣ ␣	2 em	\ : ↩	0.4 spc
␣ ␣ ␣ ␣	3 em	\ ; ↩	0.6 spc

Table 7.1. Keystrokes for common horizontal spaces.

For minor adjustments of spacing inside text or formulas, it is recommended to use one of the following length units:

spc. The width of the space character in the current font.

fn. The current font's design size with a flexibility of half that amount.

em. The (fixed) width of the "M" character in the current font, also called "quad space".

The advantage of these units is that they tend to be proportional to the font size, so your adjustments remain correct if you change this size. See Table 3.9 for more available length units.

You may use the keystrokes from Table 7.1 in order to quickly insert small horizontal spaces. The amount of space can be changed *a posteriori* using Focus ▸ Set length or the Length field on the focus toolbar. Even more efficiently, you may keep the ⌘ modifier key pressed, and then adjust using the arrow keys. We note that negative horizontal spaces are also supported; one typical application is to adjust the position of the subscript in Γ_a, and turn this formula into Γ_a.

It is sometimes useful to specify a vertical depth and height for a whitespace using Format ▸ Whitespace ▸ Rigid box. For example, the space "   " is 1 cm wide, -3 mm deep, and 5 mm high. The Format ▸ Whitespace ▸ Horizontal space menu also contains entries Tab and Custom tab for "horizontal tabs". These can be used if you need more than one type of alignment on the same line: for a line with one tab, the text before and after the tab are respectively left- and right-aligned. With two tabs, the three portions of text around the tabs are left-aligned, centered, and right-aligned. You may also insert a tab using the keyboard shortcut ⌘⇥. Custom tabs allow you to specify the minimal amount of space that will always be inserted, even when the tab occurs in a line full with text.

7.5.2 Vertical spacing

Vertical spaces can be inserted using Format ▸ Whitespace ▸ Vertical space before or Vertical space after. The markup that corresponds to such a vertical space is indicated through an "informative flag"; the space itself occurs either before or after the current paragraph. In particular, for paragraphs with more than one line, there may be several lines between the place where you make your "vertical space request" and the actual space.

Whereas $a_{b_{c_d}}$ descends below the line, the iterated exponential $\mathrm{e}^{\mathrm{e}^{\mathrm{e}^x}}$ is rather high.

Whereas $a_{b_{c_d}}$ descends below the line, the iterated exponential $\mathrm{e}^{\mathrm{e}^{\mathrm{e}^x}}$ is rather high.

Figure 7.4. Vertical spacing with and without shoving-in.

The vertical spaces really act as padding: assume that you specified a "vertical space after" of `1cm` for some paragraph and a "vertical space before" of `5mm` for the next paragraph. Then the resulting space between these two paragraphs will be the maximum `1cm` of both values. This is very useful for the design of environments that are surrounded by vertical padding, such as theorem: the mechanism ensures that the same amount of space that separates the theorem from other text is also used for the separation between two consecutive theorems. When using "hard spaces" instead of padding, the space between two consecutive theorems would become twice too large (you may have encountered this annoying behavior in LaTeX).

Adding padding around environments is so common that TeXmacs actually provides a special padded tag for this, which can be inserted using Insert ▸ Prominent ▸ Padded. The Focus ▸ Preferences ▸ Padding above and Padding below menus allow you to specify the default amount of padding. We recommend to base padded environments of your own design on the padded tag whenever possible.

If you want to insert vertical space between two specific lines (instead of paragraphs), then you may use Format ▸ Whitespace ▸ Rigid box, as described in the previous section. In this particular case, you should take `1cm` for the width of your space; the height and the depth will determine the amount of whitespace with respect to the previous and the next line.

The default vertical spacing between lines can be set via Document ▸ Paragraph ▸ Interline space. You may also specify an additional space between successive paragraphs using Document ▸ Paragraph ▸ Interparagraph space. An alternative way to indicate the start of a new paragraph is to indent its first line; in that case you may take zero for the interparagraph space and control the precise amount of indentation using Document ▸ Paragraph ▸ First indentation.

The most common length unit for vertical spacing is `fn`. Occasionally, the length unit `fns` may also be of use: the minimal and default values of `1fns` are both `0fn`, but its maximal value is `1fn`. This allows for the insertion of extra vertical whitespace on underfull pages only. Several TeXmacs styles in which paragraphs are indented use `0.5fns` as the default value for the interparagraph space. The interline space can be increased similarly, using Document ▸ Paragraph ▸ Advanced ▸ Extra interline space. The default amount of extra interline space is `0.025fns`, which corresponds to the additional stretchability of one line for every forty lines.

Inline mathematical formulas frequently extend beyond the vertical limits of the line. TeXmacs implements a special "shoving-in" mechanism that avoids unnecessarily large spaces between successive lines: see Figure 7.4. To avoid visual collisions, shoving-in is used only when the horizontal distance between descending and ascending formulas (i.e. $a_{b_{c_d}}$ and $e^{e^{e^x}}$ in the example from Figure 7.4) is sufficient. This minimal horizontal distance is a style parameter that can be modified via Document ▸ Paragraph ▸ Advanced ▸ Horizontal collapse distance. The actual vertical space between formulas always exceeds the minimal line separation space: Document ▸ Paragraph ▸ Advanced ▸ Minimal line separation.

7.5.3 Adjusting the position and extents

The insertion of whitespace is a very common technique for adjusting the layout of text and formulas. Another useful approach is to explicitly move or resize certain parts. For instance, consider the formula

$$\begin{pmatrix} 2^x & 0 \\ 0 & 2^y \end{pmatrix} + \begin{pmatrix} 3^f & 0 \\ 0 & 3^g \end{pmatrix}.$$

The fact that the character f is higher than x causes TEXMACS to use larger brackets for the right-hand matrix. In order to obtain the better-looking

$$\begin{pmatrix} 2^x & 0 \\ 0 & 2^y \end{pmatrix} + \begin{pmatrix} 3^f & 0 \\ 0 & 3^g \end{pmatrix},$$

it suffices to select the character f and “smash” it using Format ▸ Adjust ▸ Smash. The smashed f is displayed as before, but the typesetter is told to believe that its vertical extents are the same as for the character x. It is also possible to reduce the vertical extents by an explicit amount of space using Format ▸ Adjust ▸ Reduce. Inversely, you may “inflate” the character x using Format ▸ Adjust ▸ Inflate, and thereby pretend that it has the same vertical extents as the largest character in the font, e.g. f.

Further types of adjustments that you may find in the Format ▸ Adjust menu are the following:

Move. Explicitly move a subformula by both a horizontal and a vertical amount of space. This can both be used for fine-tuning and for graphical effects such as bubbling.

Shift. This is a variant of Move with this difference that the extents of the object are not altered by the shift. In other words, shifting does not alter the typesetting of any surrounding text; it just displays the shifted object at another place.

Resize. Explicitly modify the extents of a piece of text (i.e. its left, right, bottom, and top limits), without changing its rendering. The above mechanisms of smashing, inflation, and reduction are all special cases of resizing.

Extend. Similar to Resize except that we specify increments with respect to the default extents.

Clip. Similar to Resize with the difference that the piece of text is "clipped" to remain inside the specified bounding box. For instance, when clipping the text "clipped" vertically to the extents of the character "x", we obtain clipped.

When specifying the length arguments for the above constructs, you may use the length units `w` and `h` for the width and height of the original content. You may also use `l`, `r`, `b`, and `t` for the left, right, bottom, and top limits.

7.5.4 Phantoms

Phantoms are yet another typesetting technique that is occasionally useful. A piece of text can be turned into a phantom by selecting it and applying Format ▸ Special ▸ Phantom. The phantom text is treated similarly to the original text, except that it is rendered using invisible ink. Let us give two examples of how to apply this feature.

Tensors are objects that may contain subscripts and superscripts, but for which the subscripts and superscripts need to be aligned in a precise manner:

$$A_{ik}^{j} \cdot B_{j}^{il}.$$

This effect can be obtained by inserting a phantom superscript i above the subscript i of A, followed by a phantom subscript j below the superscript j, and so on.

Another example is a book on polynomial elimination, which typically contains many explicit polynomials in a finite number of variables, such as

$$x_1 x_2^6 x_3 + 5 x_1^5 x_3^3 - 7 x_2 x_3^7 - 2016 x_3^8.$$

The subtle typographic problem here is that the subscripts of x_1 and x_2^6 are not aligned, due to the fact that x_1 has no superscript (no need to move the subscript further downwards). One possible remedy is to use "phantom 1" superscripts whenever a variable occurs without any exponent:

$$x_1^{} x_2^6 x_3^{} + 5 x_1^5 x_3^3 - 7 x_2^{} x_3^7 - 2016 x_3^8.$$

Of course, it would be even better if TEX_{MACS} implemented a special style parameter that allows similar subscripts and superscripts to be placed automatically at the same level.

7.6 Miscellaneous typesetting utilities

7.6.1 Linear transformations

Several kinds of linear transformations can be applied to selected pieces of text using the items in Format ▸ Transform: rotations (Rotate), horizontal and vertical resizing (Dilate), slanting (Skew), and general 2×2 matrix transformations. The corresponding parameters (angle, magnification factors, slant, and matrix coefficients) can be entered using the fields on the focus toolbar. Notice

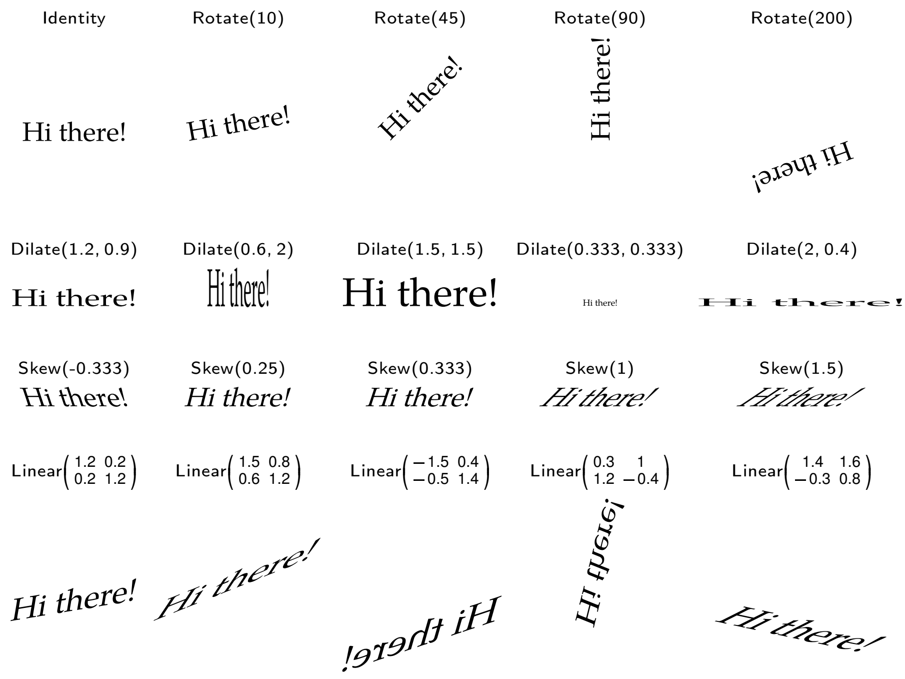

Identity
Rotate(10)
Rotate(45)
Rotate(90)
Rotate(200)
Dilate(1.2, 0.9)
Dilate(0.6, 2)
Dilate(1.5, 1.5)
Dilate(0.333, 0.333)
Dilate(2, 0.4)
Skew(-0.333)
Skew(0.25)
Skew(0.333)
Skew(1)
Skew(1.5)
$\text{Linear}\begin{pmatrix}1.2 & 0.2\\ 0.2 & 1.2\end{pmatrix}$
$\text{Linear}\begin{pmatrix}1.5 & 0.8\\ 0.6 & 1.2\end{pmatrix}$
$\text{Linear}\begin{pmatrix}-1.5 & 0.4\\ -0.5 & 1.4\end{pmatrix}$
$\text{Linear}\begin{pmatrix}0.3 & 1\\ 1.2 & -0.4\end{pmatrix}$
$\text{Linear}\begin{pmatrix}1.4 & 1.6\\ -0.3 & 0.8\end{pmatrix}$
Hi there!

Figure 7.5. Applying several linear transformations to the text "Hi there!".

that linear transformations can be applied to any kind of content, both text and graphics. In sections 7.7 and 7.8 below, we will discuss several transformations and effects that are applied more specifically to text or graphics. Figure 7.5 demonstrates various common linear transformations.

7.6.2 Device dependent rendering

Informative flags indicate the positions of labels or typesetting directives while editing, but they are removed when printing the document on paper. This is one example of when it is useful to differentiate between rendering on the screen and on paper. More generally, the Format ▸ Specific menu can be used to make some portion of text visible only under specific circumstances. The following filters are available:

Screen. Only show the content on the screen and not when printed.

Printer. Only show the content when printing it on paper, but not on the screen.

LaTeX. Only show the content when exporting to LaTeX [38].

Html. Only show the content when exporting to HTML [41].

TeXmacs. Show the content except when exporting to another format.

Image. When exporting this content to another format, export it as an image.

Odd pages. Only show the content on odd-numbered pages, corresponding to the right-hand side pages in a book.

Even pages. Only show the content on even-numbered pages, corresponding to the left-hand side pages in a book.

7.6.3 Repeating a pattern

In tables of contents, one often uses horizontal dots to connect the section titles and the corresponding page numbers. This is done by repeatedly inserting the same pattern (a dot) over the full width of a given portion of text (in this case a "horizontal tab" between a section title and its page number). Similarly, by repeating a slash "/", you may obtain the effect of ~~stricken out~~ text.

You may create your own repetitions of text using Format ▸ Special ▸ Repeat object. The second argument of the repeat tag contains the pattern that you wish to repeat. The first argument corresponds to the text on top of which the repetition is applied. For instance, ⟨repeat|the answer|.⟩ yields the answer. For obtaining the effect of horizontal dots in a table of contents, you should use Format ▸ Whitespace ▸ Tab as the first argument.

7.6.4 Line breaking of transformed text

Line breaking is naturally compatible with changes in the current font and text color: if you put a large blue italic citation in the the middle of some paragraph, then TeXmacs may line break it just the same as ordinary text.

Although blurring and boxing are similar to changing the font or text color in many ways, these transformations can only be applied to contiguous portions of text. In other words, the usual line breaking algorithm will not work for blurred or boxed texts.

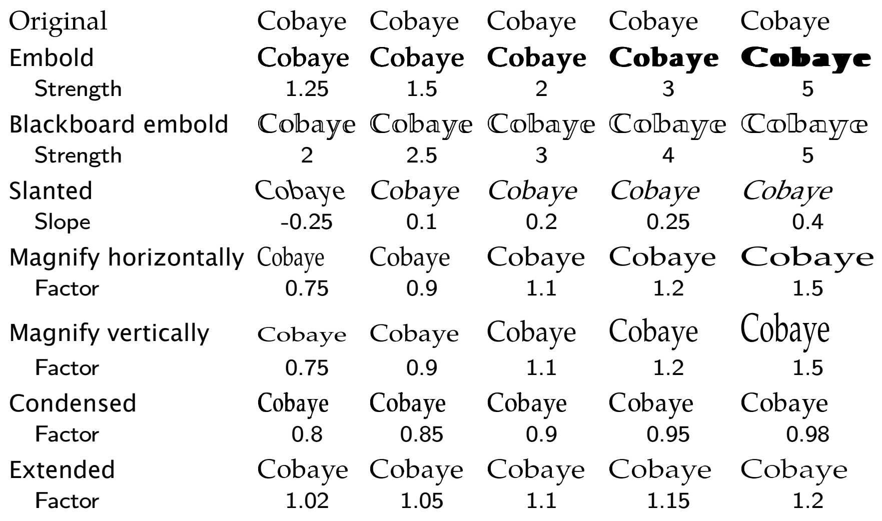
In order to extend line breaking to such cases, TeXmacs provides Format ▸ Special ▸ Decorate atoms. The first argument of the datoms primitive should be a unary macro for transforming the lines of a broken paragraph. The second argument contains the text to which the transformation should be applied. For the above types of transformations, this allows us to obtain

> This is a sample text that should clarify things, even some of the very vague ideas that spur out of my blurry mind, as well the more important thoughts that might be worth being emphasized very loudly.

In this example, we notice that the boxed text is somewhat too wide: the mechanism only works optimally if transformed lines of text are exactly as wide as the untransformed ones.

7.7 Font effects

So far, we have described numerous tricks to increase your control over the typesetting process. TeXmacs also provides several graphical effects for giving your documents a more artistic twist.



Original	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Embold	**Cobaye**	**Cobaye**	**Cobaye**	**Cobaye**	**Cobaye**
Strength	1.25	1.5	2	3	5
Blackboard embold	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Strength	2	2.5	3	4	5
Slanted	Cobaye	*Cobaye*	*Cobaye*	*Cobaye*	*Cobaye*
Slope	-0.25	0.1	0.2	0.25	0.4
Magnify horizontally	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Factor	0.75	0.9	1.1	1.2	1.5
Magnify vertically	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Factor	0.75	0.9	1.1	1.2	1.5
Condensed	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Factor	0.8	0.85	0.9	0.95	0.98
Extended	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Factor	1.02	1.05	1.1	1.15	1.2

Figure 7.6. Basic font effects for various values of their corresponding parameters.

A first family of graphical effects, available in Format ▸ Font effects, operates on the font glyphs. The purpose of many of these effects is to mimic common font styles such as **bold**, *slanted*, extended, condensed, etc. It is important to note that these imitations are rather poor substitutes for their hand-designed counterparts. There are many reasons for this. For instance, in most font families, italic characters such as *a*, *f*, and *g* are completely different from their slanted counterparts *a*, *f*, and *g*; even other characters such as *b* and *l* differ in many ways from *b* and *l* (e.g. the lower corner of the "b" and its aspect ratio, the serifs of the "l", etc.). Other shortcomings of machine imitations are more subtle and typically concern the stroke widths and internal proportions.

Despite the above limitations, there are often no good alternatives to machine imitations. If you need a font style that simply does not exist for a given font family, then your only remaining option is to find a similar font that does provide the required style. The desired quality of a particular font may also depend on the purpose: a poor counterfit may be acceptable for the occasional use in a laptop presentation, but becomes more problematic when you systematically need it inside a reference manual.

TeXmacs automatically tries to find a suitable font when you change the font style, but the artificial font effects from Format ▸ Font effects give you additional control. In the case of an emboldened font, you may for instance specify the emboldening factor on the focus toolbar. This gives rise to a whole range of weights such as "semibold", "bold", "black", "heavy", "ultra-black", etc. Figure 7.6 shows imitations of various common font effects for different values of their parameters.

TeXmacs proposes two different mechanisms to produce extended and condensed fonts. The first one proceeds by horizontal magnification, but has the drawback that the pen width of vertical strokes changes. The second mecha-

Factor	0.75	0.9	0.95	0.98	1.02	1.05	1.1	1.25
I	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
II	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
I/II	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye

Figure 7.7. Various techniques for emulating condensed and extended fonts.

nisms uses a sophisticated algorithm in order to preserve the pen width. However, this results in a lower “ink density” for wider fonts. For high quality emulations of condensed and extended fonts, a good compromise might be to use a combination of both mechanisms; see Figure 7.7 for a comparison.

Besides emulations of common font styles, $\text{T}_\text{E}\text{X}_\text{MACS}$ also implements a few more artistic font effects: see Figure 7.8. These effects involve random, so-called *Perlin noise*. The Degraded effect conserves the general shape of glyphs, but randomly removes little regions of pixels. The size of these regions is controlled using the “frequency” parameter, whereas the amount of degradation depends on the “threshold” parameter. You may use Distorted in order to randomly distort glyphs at their boundaries. Such distortions work “both ways” in the sense that ink may leak to the outside or be moved further inside. If you don't want to permit leaks to the outside, then you may use the Gnawed effect instead.

It is worth pointing out that the control parameters of all font effects are environment variables. As such, you may both modify them locally, using the fields on the focus toolbar, or globally, via Focus ▸ Preferences ▸ Style parameters. For example, assuming that your cursor is inside a degraded tag, you may change the default degradation threshold from 0.667 into 0.6 using Focus ▸ Preferences ▸ Degraded threshold ▸ Other.

We also note that many of the above font effect rely on low-level bitmap operations on the font glyphs. This means in particular that the transformed fonts are no longer vectorial “Type 1” fonts, but only “Type 3” bitmap fonts. Some publishers prohibit PDF files that include such fonts. This drawback might vanish in future versions of $\text{T}_\text{E}\text{X}_\text{MACS}$ with the development of vectorial counterparts for all font effects. Font emboldening, slanting, and magnification are already vectorial.

Original	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Degraded	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Threshold, Frequency	0.5, 1	0.6, 1	0.667, 1	0.75, 1	0.667, 0.5	0.667, 2
Distorted	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Strength, Frequency	0.25, 1	0.5, 1	0.75, 1	1, 1	0.5, 0.5	2, 2
Gnawed	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye	Cobaye
Strength, Frequency	0.3, 1	0.6, 1	0.9, 1	1.2, 1	0.6, 0.5	2, 2

Figure 7.8. Artistic font effects.

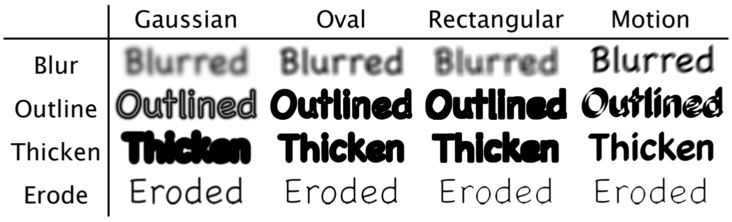


Figure 7.9. Basic graphical effects using different pen styles.

7.8 Graphical effects

A variety of graphical bitmap effects are available in the menu Format ▸ Graphical effects. They all operate on rectangular regions of text by first transforming them into bitmap pictures and then applying the graphical effect. For large pieces of texts, these transformations may take some time, so it is recommended to apply the effect after you are done editing the text.

Several graphical effects like "distortion" are also available as font effects. There are a few differences though. First of all, we recall that font effects operate on individual glyphs. $\text{T}_\text{E}\text{X}_\text{MACS}$ in particular remembers the transform of every glyph. In the word "distorted", this implies that the two letters "t" are distorted in exactly the same way. In the graphical counterpart "distorted", multiple occurrences of the same character are distorted differently. The fact that $\text{T}_\text{E}\text{X}_\text{MACS}$ remembers transformed glyphs has the additional benefit of speed: the transformed font essentially operates at the same speed as the native one. Another important difference between font effects and their graphical counterparts is that the first may also adjust the metric properties of the glyphs. Consider the following **emboldened** characters. The analoguous graphical effect of "thickening" with a "motion pen" again produces **emboldened** text, but with a reduced spacing.

As in the case of font effects, each of the graphical effects comes with a certain number of control parameters that can be modified in the focus toolbar. Four basic effects are blur, outline, thicken, and erode. The precise action of these effects is determined by the *pen* that you use. Currently, four types of pens are supported in Focus ▸ Effect pen: Gaussian, Oval, Rectangular, and Motion. This menu is also available on the icon toolbar, under the icon that shows the current pen: ●, ●, ◆, or ⟋. For each of the pen styles, you may specify the width, the height, and an angle. In Figures 7.9 and 7.10, you may see the results obtained with various pens.

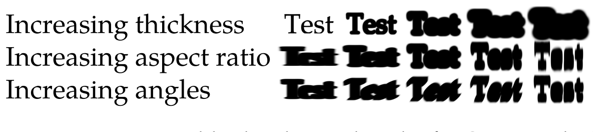


Figure 7.10. Various pen widths, heights, and angles for Gaussian thickening.

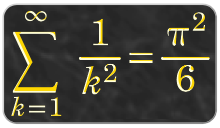


Figure 7.11. Applying an engraving effect to a formula.

The next group of effects is built on top of the above more basic effects through superposition, intersections, and recoloring. The shadow effect adds a Gaussian drop shadow to the text in a color that can be specified by the user. The shadowed raise effect is a variant in which the roles of the shadow and the main text are interchanged. Alternative ways to "raise" or "emboss" text are emboss and outlined emboss. The opposite effects are engrave and outlined engrave. Engravings require the colors and pen parameters to be chosen with care, as in Figure 7.11, where we put a golden formula on a granite background, using "pastel yellow" for the shadow color.

The random effects degrade, distort and gnaw are similar to the corresponding font effects. One advantage of the graphical variants is that they can be applied to text as well as pictures: see Figure 7.12.

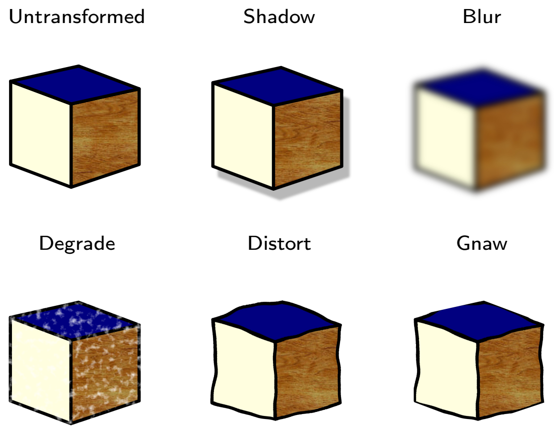


Figure 7.12. Applying graphical effects to pictures.

Chapter 8
Drawing pictures

8.1 Introduction

One of the simplest ways to insert pictures into TeXmacs documents is to save your picture in a separate file and then use Insert ▸ Image ▸ Link image or Insert image. Most of the standard file formats for images are supported, which allows you to insert both photographs and pictures that were drawn using externals tools such as GIMP, INKSCAPE, or DIA [35, 58, 39]; see section 6.3.

TeXmacs also includes a tool for creating your own drawings directly from within the editor. Whether this tool should be preferred over external software depends on the circumstances. Programs dedicated to image manipulation typically contain sophisticated features for obtaining specific graphical effects. However, these features are often designed for professional artists, so the learning curve may be steep for a casual user. If you only need to draw fairly simple objects such as lines, curves, text, arrows, etc., then elaborate special-purpose software may be overkill.

Using the TeXmacs picture editor also offers several advantages. First of all, its integrated nature makes it easy to insert rich text, mathematics, and hyperlinks inside your pictures. Mathematical formulas are often problematic for externals tools. Secondly, native images often look nicer, because they use the same fonts and default line width as the surrounding text. Unless you really need specific graphical constructs that are not provided by TeXmacs, native pictures should therefore be preferred if quality is an issue. Finally, as long as you only need basic graphical elements, the integrated picture editor should be reasonably complete and user-friendly.

8.2 Starting new pictures

New pictures can be started using Insert ▸ Image ▸ Draw image and they are edited in *graphics mode*. By default, a newly inserted image extends over the whole paragraph. You may adjust its size using the keyboard shortcuts ⌥←, ⌥→, ⌥↑, ⌥↓ (to adjust the size a bit faster, you may use ⌥⇧←, ⌥⇧→, ⌥⇧↑, ⌥⇧↓). An explicit size can also be specified using Insert ▸ Geometry ▸ Size or 🪄 ▸ Size. After completion of your drawing, it is possible to automatically crop it to its actual size (plus some additional padding), using Insert ▸ Geometry ▸ Crop or 🪄 ▸ Crop; an example of such a cropped image is shown in Figure 8.1.

In some cases, you may want to draw something on top of an existing image or other kinds of content. This can be done by selecting the relevant background image and clicking on Insert ▸ Image ▸ Draw over selection. Similarly, you may use Insert ▸ Image ▸ Ink here to make manual graphical annotations at

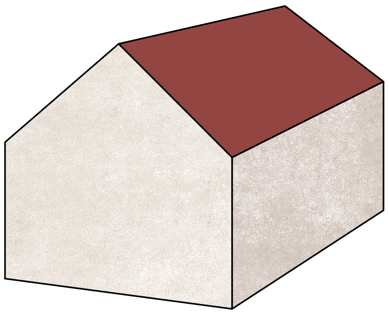
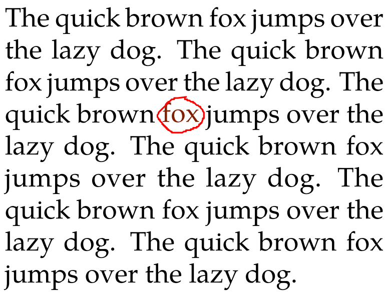


Figure 8.1. Example of a "cropped image" with a default padding of `1spc`.



The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

Figure 8.2. Example of inking above the selected brown fox.

the current cursor position using the mouse (or a pen in the case that you have a tablet).

The following formula shows an example of drawing a picture over an existing formula:

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

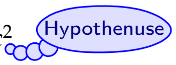


Figure 8.2 shows a similar example in which the red oval was inked over the text. When making laptop presentations, it can be extremely useful to put graphical and mathematical annotations on top of existing pictures: see Figure 8.3.

By design, graphical annotations (created using Draw over selection or Ink here) are drawn on top of the annotated text. Annotations are also allowed to partially overlap with the surrounding text. The amount of permitted overlap can be specified using Insert ▸ Geometry ▸ Overlap. This basically determines the region where you are allowed to draw.

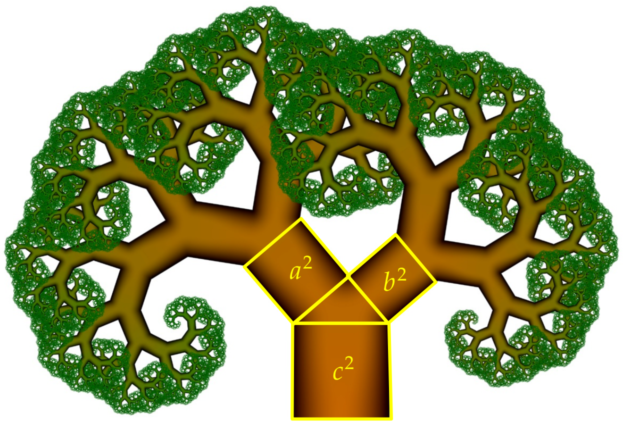


Figure 8.3. Picture of a Pythagoras tree with mathematical annotations. The picture is due to Atze van der Ploeg and freely available from WIKIMEDIA.

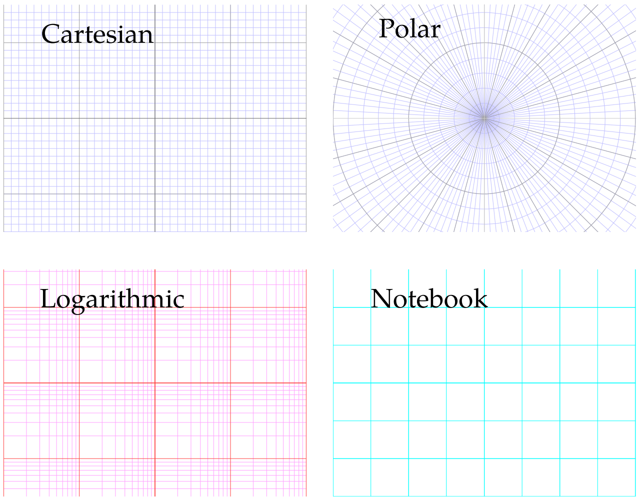


Figure 8.4. Various available types of grids with different colors.

You may exit the graphics mode using Focus ▸ Exit graphics or . You may re-enter the graphics mode by clicking inside the annotated text; alternatively you may put your cursor just behind the annotated text and use Focus ▸ Enter graphics or . The annotated text itself can be edited by replacing the draw-over tag by a draw-under tag using Focus ▸ Draw over ▸ Draw under (or Insert ▸ Geometry ▸ Draw over from inside the graphics mode).

Inside graphics mode, the cursor is replaced by a red cross + so as to indicate the current position more precisely. For even more control, it is often useful to display a grid while you are drawing (see Figure 8.4), which can be done using Insert ▸ Grid ▸ Type ▸ Cartesian or ▸ Type ▸ Cartesian. In that case, the red cross is automatically "snapped" to the grid as soon as you approach grid points or edges. This makes it easy to draw pictures that contain many lines between grid points, or other graphical elements whose "control points" lie on a grid. Snapping can be useful on other occasions as well, such as starting a line segment at the exact intersection of two curves; see section 8.7 for more details.

In the menu Insert ▸ Grid or it is possible to adjust the colors of the axes and the grid-lines, as well as the number of subunit grid-lines per unit grid-line. By default, grids are printed; you need to remove them after completing your drawing if you do not want them to appear in print.

The origin of the grid is initially set at the center of the screen. You may scroll the picture using the arrow keys ←, →, ↑, ↓ (or ⇧←, ⇧→, ⇧↑, ⇧↓ if you want to move fast). The default unit for grids is 1 cm. You may specify a different unit using the menu Insert ▸ Geometry ▸ Unit or ▸ Unit. You may also zoom in and out using + and -, Insert ▸ Geometry ▸ Zoom, or ▸ Zoom.

8.3 Graphics mode

When drawing pictures, the user interface of TEXMACS works in a slightly different way than in text mode. Before going into more details, let us briefly summarize the main design philosophy behind the graphical editor.

We have already encountered the icons and for specifying the global properties of the image and of a guiding grid.

The second and third groups of icons , , , , , , , T, , , ∝ on the mode-dependent toolbar can be used to insert new graphical elements. For instance, clicking on will put the editor into "insert polyline mode". This allows you to insert new polyline objects: one mouse click for the first and every next point, and a double click for the last point. The vertices of such a polyline object are also called *control points*. You may move control points of existing polyline objects by dragging them with the mouse. In "insert polyline mode" it is also possible to move control points of circles and other graphical elements in this way; this saves you the trouble of switching modes for such basic *a posteriori* adjustments.

TEXMACS supports various kinds of curves such as polylines, arcs of circles, splines, and Bezier curves. In each of these cases, the shape of the curve is specified via its control points, which typically (but not necessarily) lie on the curve itself. Other graphical elements may also contain text. The most basic textual objects consist of one control point for the position and the corresponding text itself. The more elaborate "arrow with text" objects consist of two control points for the start and the end of an arrow, together with accompanying text. Such objects may for instance be useful for drawing commutative diagrams. You may enter "insert text mode" using T, whereas arrows with text are only available via the Insert menu.

The rendering of a graphical object is completely determined by its control points, its text fields (if any), and its *style properties*. The available style properties for a given mode are shown in the focus toolbar. The style properties of graphical objects can also be edited via the focus toolbar. In "insert line mode", this allows you to select a color, a line width, possible arrows at the endpoints, etc. For textual elements, the style properties typically include horizontal and vertical alignment. Using the ↩ key, it is also possible to apply the current style properties to the object under your mouse. Conversely, the shortcut ⇧↩ allows you to fetch the properties of an existing object. This may be useful if you want to apply the same properties to another object.

The last group of icons , , , , on the mode-dependent toolbar can be used to edit existing objects or groups of objects. The first icon allows you to modify style properties. The next three icons , , make it easy to move, resize, and rotate objects. The last icon is used for grouping a collection of graphical objects inside a single object and *vice versa*.

Technically speaking, it should be noted that a graphical picture is simply a list of graphical objects that are drawn on top of each other in sequence. The order in which this is done is called the *stacking order*. Whenever several objects overlap, the stacking order clearly influences the rendering of a picture. Inside graphics mode, when hovering the mouse over an object, you may change its stacking order using the ⇞ and ⇟ keys: ⇞ raises your object "one level up" in front of other objects, whereas ⇟ moves it "one level down".

We also note that some of the actions in graphics mode use the right mouse button. If your mouse has only one button, then it is usually possible to emulate the right button by holding the control key while pressing the unique mouse button.

8.4 Graphical objects

Points

Points are among the simplest objects supported by $\mathrm{T_EX_{MACS}}$. You enter "insert point mode" using or Insert ▸ Point. New points are inserted via a single mouse click and existing points can be moved by dragging them using the mouse. Points can be removed by pressing ↩ or the right mouse button while hovering over them. The color, shape, and size of points can be specified using the style properties.

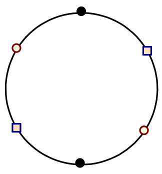


Figure 8.5. Six points on a circle.

Polylines and closed polygons

You enter "insert polyline mode" using or Insert ▸ Line. In this mode, a new polyline (also called a broken line object) can be started via a single mouse click. At every next mouse click, a new control point is added. The last control point is specified using a double click. While entering a polyline, you may use the right button of your mouse in order to remove the latest added point. This is equivalent to the "undo" shortcut ⌘[.

You may also edit existing polyline objects inside "insert polyline mode" or any other "insert mode". We already mentioned that control points can be moved by dragging them using the mouse (Figure 8.6b). In addition, a new

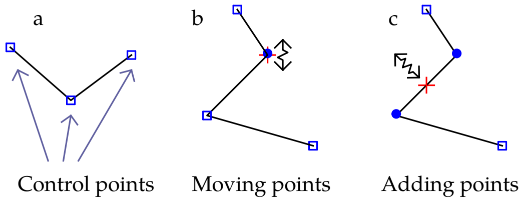


Figure 8.6. *A posteriori* editing of polylines.

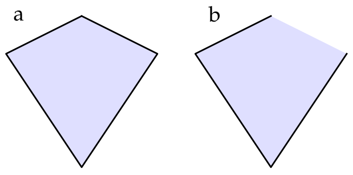


Figure 8.7. A filled polygon and polyline.

control point can be inserted on an existing line segment using the same technique. In order to do so, hover the mouse pointer over the edge where you want to insert a new point; the two neighboring control points should be highlighted (Figure 8.6c). Then start dragging and a new control point will appear. When hovering over a control point, the right mouse button or the ⌫ key can be used to remove it. When holding the shift key ⇧ while clicking on the right mouse button, the entire object will be removed.

Polygons are a variant of polylines for which the last and the first control point are also joined by a line. You may draw them using or Insert ▸ Polygon. The style properties allow you to specify a fill color or pattern for the interior of a polygon (Figure 8.7a). Broken line objects may also be filled, but the result usually looks less nice (Figure 8.7b).

Circles and arcs of circles

You may enter "insert circle mode" using or Insert ▸ Circle. New circles are inserted by specifying three points on the circle using single mouse clicks. Arcs of circles are inserted in a similar way, provided that you are in "insert arc mode" (or Insert ▸ Arc). As usual, the three control points can be moved *a posteriori* by dragging them using the mouse. The style properties allow you to specify a fill color (Figure 8.8).

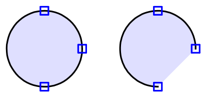


Figure 8.8. A filled circle and a circular arc, together with their three control points.

Smooth curves

Smooth curves are similar to polyline objects with this difference that they are nicely rounded at the control points. You may draw smooth curves using or Insert ▸ Curve; their closed variants can be obtained through or Insert ▸ Closed curve. New curves and control points can be inserted in the same way as for polyline objects and polygons. Removal and *a posteriori* editing of control points also works similarly.

TeXmacs supports two types of smooth curves: *splines* and *cubic Bezier curves*. Splines are the default type of curves for and . The precise shape of a spline curve is determined as a function of all control points. Moving a control point will therefore affect the whole curve and not just the neighboring parts, even though the changes will be most prominent near the control point. By contrast, the shape of individual pieces of a Bezier curve is entirely determined by the four closest control points.

The differences between splines and Bezier curves can be seen in Figures 8.9a and 8.9b. The banana from Figure 8.9a is a closed spline with four control points. Figure 8.9b shows the closed Bezier curve that is obtained by using the same control points.

The banana from Figure 8.9b is actually a "simplified" Bezier curve, that was inserted using Insert ▸ Closed curve ▸ Closed smooth. More general (cubic) Bezier curves have two types of control points: principal control points that lie on

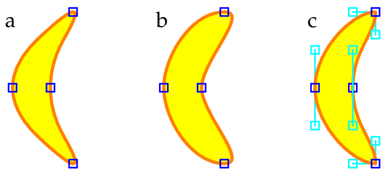


Figure 8.9. Comparing splines and Bezier curves.

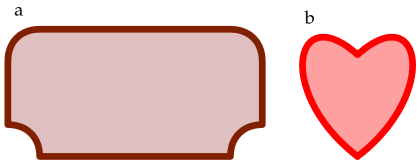


Figure 8.10. Bezier curves with general tangents.

the curve and additional control points that specify the tangents of the curve at these principal control points. Every principal control point is surrounded by exactly two control points for controlling these tangents. In Figure 8.9c we have drawn yet another banana by using such a general Bezier curve. Notice that we specified both a horizontal and a vertical tangent at each of the two endpoints of the banana.

General Bezier curves are inserted using `Insert ▸ Curve ▸ Bezier` or `Insert ▸ Closed curve ▸ Closed Bezier`. They are a bit more awkward to edit because of the different types of control points. But they are very useful for technical drawings that require very precise control over the curves. For instance, it becomes possible to alternate straight and rounded portions within the same curve (Figure 8.10a). They also allow you to draw heart-like shapes (Figure 8.10b).

Text objects and mathematical formulas

Ordinary text can be incorporated into pictures after switching to "insert text mode" (T or `Insert ▸ Text`). In this mode, a mouse click inserts a text object at that position. While editing text, you are no longer in graphics mode, and the interface will change to the usual one for editing text. In order to return to the graphics mode, it suffices to click anywhere inside the picture. In case your picture contains several text objects, you may easily move from one object to the others using the cursor keys ←, →, ↑, and ↓. As usual, text objects can be moved by dragging them using the mouse and removed using the right mouse button or ⌫.

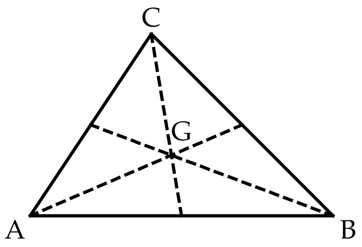


Figure 8.11. Example of text in a picture.

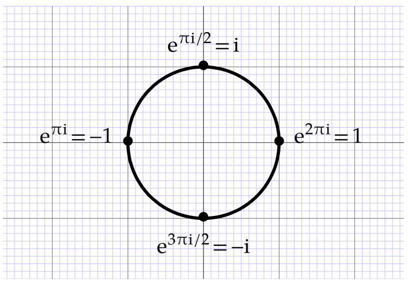


Figure 8.12. A picture with mathematical formulas.

TEXMACS also has a special mode for inserting mathematical formulas (ξ_n^2 or Insert ▸ Mathematics). The same graphical result can be obtained by inserting a text object as above and then creating a mathematical formula inside it using $. Nevertheless, the dedicated "insert formula mode" is handy for pictures in which you need to insert many formulas.

Longer blocks of text

The above "insert text" and "insert formula" modes are mainly intended for short fragments of text. For longer portions of text, the "insert text block" mode is more appropriate (▤ or Insert ▸ Long text). Line wrapping is activated in this mode and text blocks are allowed to extend over several paragraphs. Using style properties that are available from the focus toolbar, you may control the width of text blocks, as well as the interline spacing and optional borders and background colors.

Hand-drawn curves

Sometimes it may be desirable to quickly sketch a picture as if you were drawing it using a pen on a sheet of paper. This is particularly natural if you own a tablet PC or if you connected an external graphical tablet to your computer. The graphical editor offers a special "inking mode" to this purpose, which is entered using ∝ or Insert ▸ Hand drawn.

If you own a graphical tablet with a pen, the "inking mode" allows you to directly draw curves on the screen. It is also possible to "ink" using a traditional mouse by holding the mouse button down while drawing. Currently, the inking mode produces straightforward polylines that can be edited *a posteriori* as described above. Notice however that inked curves usually contain many control points, which makes it tedious to edit them one by one. On the other hand, global transformations are easy to apply *a posteriori*, as explained in section 8.6.

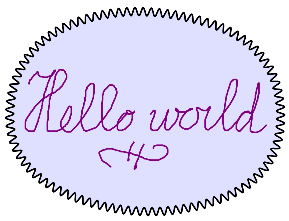


Figure 8.13. Hand-drawn greeting.

Other graphical objects

In addition to the primitive graphical objects described above, TEXMACS also provides a few graphical macros for drawing more complex objects that are composed of primitive objects. For the moment, these graphical macros are still somewhat experimental, so they have only made available from the Insert menu. Currently, the "arrow with text" macro is the most important one, since it may be useful for the creation of commutative diagrams: see section 4.8. The experimental `graphical-macros` package contains several macros for elec-

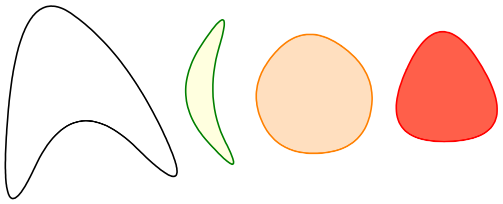


Figure 8.14. Closed splines with various line and fill colors.

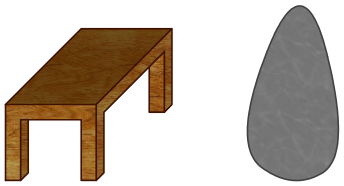


Figure 8.15. Pattern fill demo.

tronic components, among others. Future versions of $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ are likely to provide more and more constructs of this kind.

8.5 Style properties

Color

This property applies to any of the graphical object types and specifies the line or font color. The color can either be selected from a fixed palette of common colors, using a special color picking window (Focus ▸ Color ▸ Palette), or by name (Focus ▸ Color ▸ Other). $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ supports many familiar colors by their English names, such as red, purple, gold, or olive. In addition, you may use hexadecimal values of the form `#rrggbb`, such as #c04080 (75% red, 25% green, 50% blue).

Fill color

This property applies to all graphical object types except text and mathematics. It specifies a fill color for the object. Notice that $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ also allows you to specify a pattern for the fill color (see Figure 8.15).

Opacity

This property applies to all graphical object types and specifies an opacity between 0% and 100%. The default is 100% and lower opacities will make the object more transparent (see Figure 8.16).

Line width

The line width property applies to all curves (polylines, splines, Bezier curves, arcs, circles, and their closed variants), including curves that may occur inside other graphical objects (such as arrows with text). The default line width is `1ln`, which corresponds to the width of the fraction bar in mathematical formulas. Figure 8.17 shows the effect of changing the line width. It is recommended

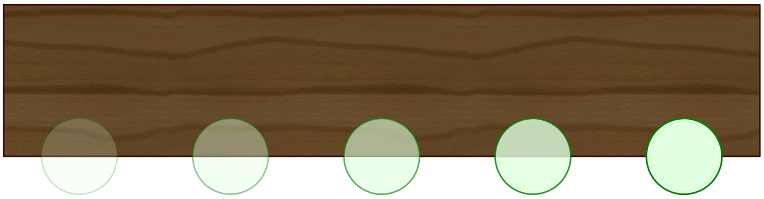


Figure 8.16. A circle on top of another object, using increasing opacities.

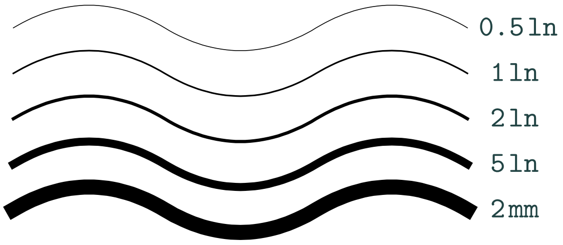


Figure 8.17. The same curve using different line widths.

to use `ln` as a unit for line widths, but any other length unit such as `mm` is also allowed.

Dash style and dash patterns

Various *dash styles* are available for curves in Focus ▸ Line dashes (see Figure 8.18). In addition to the standard dash styles, $\TeX_{\text{MACS}}$ offers several two-dimensional *dash patterns*: see Figure 8.19.

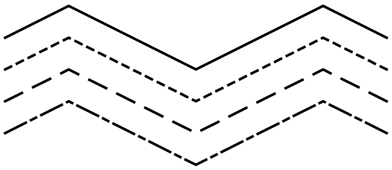


Figure 8.18. Dash styles.

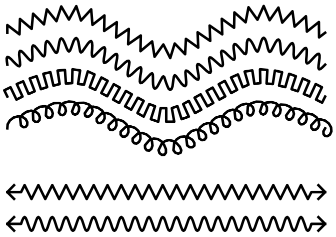


Figure 8.19. Dash patterns.

Dash style unit

In Focus ▸ Line dashes ▸ Unit, you may specify a unit for the length of line dashes. The default unit is `5ln` and higher values result in longer dashes. $\TeX_{\text{MACS}}$ adjusts the lengths of dashes so that curves always consist of an integer number

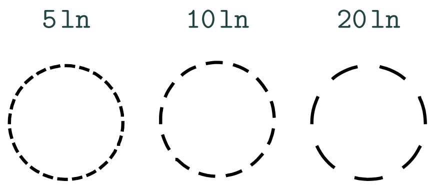


Figure 8.20. Varying the horizontal dash unit.

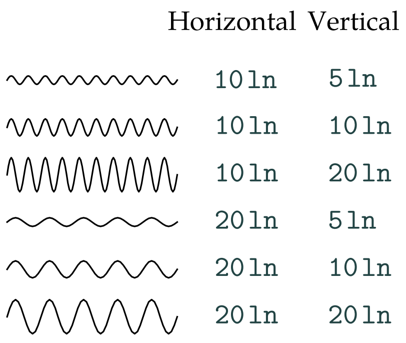


Figure 8.21. Varying the pattern dash units.

of dashes. This makes dashed curves look better, especially when they are closed or when they have arrows: see Figure 8.20.

In the case of two-dimensional dash patterns, you may specify both a horizontal and a vertical dash unit using Focus ▸ Line dashes ▸ Unit ▸ Other. In the case of "wave pattern dashes", the horizontal and vertical unit respectively control the wave length and the amplitude: see Figure 8.21. In Figure 8.22, we have also shown an example of "meander pattern dashes", with `30ln` for both the horizontal and vertical dash units.

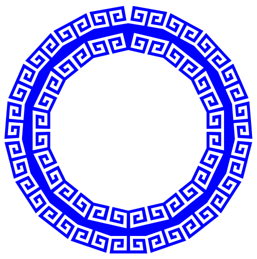


Figure 8.22. Meander pattern dashes.

Line arrows

Several kinds of arrows can be attached to the endpoints of curves using Focus ▸ Line arrows (see Figure 8.23). The orientation of such arrows is determined by the tangents to the curves at these endpoints.

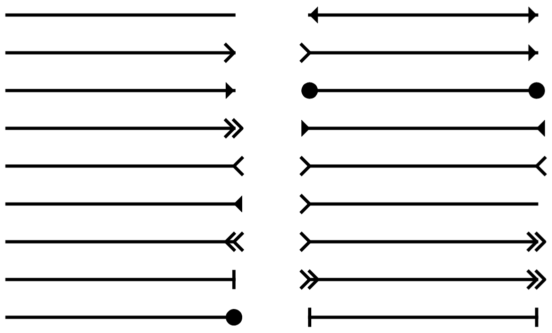


Figure 8.23. The same segment using different types of "arrows" at the extremities.

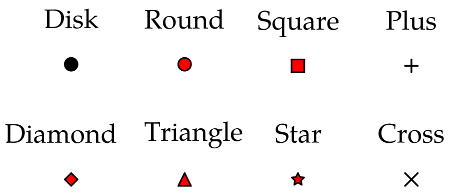


Figure 8.24. Point styles for black line and red fill colors.

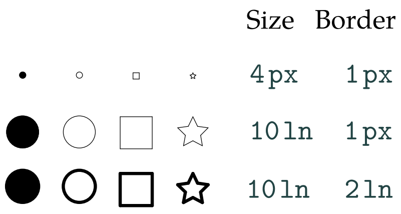


Figure 8.25. Point size and border width.

Point style

A few different styles are supported for displaying points: solid disks, circles, squares, diamonds, triangles, stars, and two types of crosses; see Figure 8.24. Two additional style properties control the size and the line width for drawing the borders. Notice that the `px` unit (which stands for the size of a pixel on the screen) remains constant while zooming in or out. The `ln` unit is proportional to other standard units such as `cm`, so that you may prefer this unit if you want the points to automatically adjust together with other graphical objects.

Text alignment

For textual and mathematical boxes, it is possible to specify their horizontal and vertical alignment. The three modes for horizontal alignment are "left", "center", and "right"; see Figure 8.26a. Besides the classical "bottom", "center", and "top" modes, TEXMACS also provides modes for vertical alignment at the baseline and axis; see Figure 8.26b. We recall that the axis is an invisible horizontal line that goes through the vertical center of the character "x".

8.6 Editing groups of objects

The "group edit" modes (which correspond to the icons , , , , on the mode-dependent toolbar) allow you to modify existing graphical objects individually or jointly. In each of these modes, the left mouse button is used to perform specific actions, whereas the right mouse button allows you to specify groups of objects on which to operate.

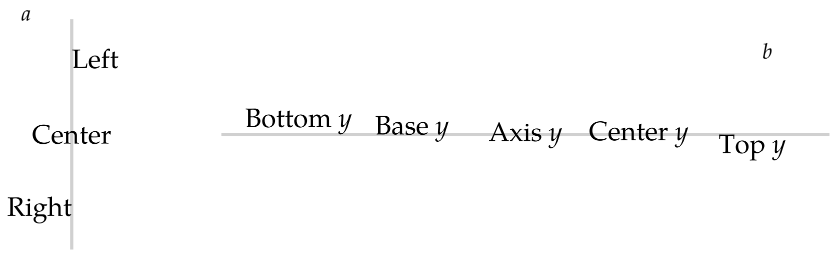


Figure 8.26. Horizontal and vertical alignment of text boxes.

If you wish to operate on a group of objects, then you may use single clicks on the right mouse button in order to toggle those objects that should be part of the group. Alternatively, you may specify a rectangular region by dragging the mouse while holding the right button down. In that case, all objects inside the region will be included in the group.

The first group editing mode is entered using or Insert ▸ Set properties, and can be used for modifying the style properties of one or more objects. If you select a group of objects, then the current properties of the group are shown in the focus toolbar. You may then modify them there or via the focus menu. If several objects of the selection have different values for a given property, then the focus toolbar will indicate "mixed" for that property. In the "set properties" mode, a left mouse click will simply select the object under your mouse.

The next three modes , , (or Insert ▸ Move objects, Resize objects, Rotate objects) allow you to move, resize, and rotate graphical objects, respectively. If you selected a group of objects, then the desired transformation can be achieved by dragging the mouse while holding the left button down. If you did not select any group of objects, then individual objects can be transformed in a similar way. In the cases of resize operations and rotations, the center of the bounding box for the transformed group of objects remains fixed.

The last group editing mode is entered using or Insert ▸ Group/ungroup. Assuming that you selected a group of several objects, a left mouse click groups these objects together into a single object. Inversely, when clicking on the resulting "group object", it is ungrouped back into simpler objects that can be manipulated individually.

The group editing modes can also be used in order to copy and paste graphical objects. Assuming that you selected a group of objects, you can copy it to the clipboard using ⌘w, , or Edit ▸ Copy. When pasting this selection using ^y, , or Edit ▸ Paste, a duplicate group will be inserted *at exactly the same location* as the original group. The new copy is automatically selected as a group, which allows you to immediately apply a transformation to it. This is particularly convenient in "move objects" mode, in which case you typically move the pasted group of objects to a new location. If you need several copies at different locations, the same procedure (pasting and moving to a new location) can be repeated.

8.7 Snapping

For technical pictures, exactness is important. When drawing freely by hand or using the mouse on a modern high precision screen, it is not so easy to specify control points up to the pixel. This makes it hard to draw lines that are exactly horizontal or vertical. Similarly, it is difficult to make a new line start at the exact intersection of two curves. What is more, such intersections usually do not have coordinates that are integer multiples of a pixel. Consequently, what may appear to be the right intersection point on your screen may look awful when zooming in or printing out on a high resolution printer.

Snapping is a technique that makes it easier to draw pictures in which lines are guaranteed to be horizontal or vertical when needed, and for which various other graphical requirements can be met exactly. The idea is that certain points, such as control points, grid points, or intersections of curves, are considered to be "special" by the editor. Whenever you approach the cursor sufficiently close to such a point, the cursor snaps to it, as if attracted by magic force.

Some of these special points may actually be more special than others. For instance, a point on a curve is more special than a generic point and the intersection of two curves is more special than a generic point on one of the intersecting curves. When two special points are competing for attention, the cursor will be snapped to the most special one. Sometimes, ambiguities still occur. In that case, you may use the ⇥ key in order to cycle among all possibilities. For instance, assume that we drew several curves starting at the same point. If we want to move the starting point of one of these curves, by dragging it using the mouse, then we may first need to hit ⇥ one or more times to make sure that we are adjusting the right curve.

The precise types of points that are considered "special" by the editor can be controlled by toggling the items in the Snap menu on the focus bar. The active kinds of snapping are indicated by the icons next to this menu; they can be deactivated by clicking on the corresponding icons. TeXmacs currently implements the following kinds of snapping (see Figure 8.27):

None. Deactivate all forms of snapping (except snapping to control points).

All. Activate all forms of snapping.

Grid points. ( ) Snaps to all grid points. For a standard Cartesian grid, this means that we snap to points (x, y) for which both x and y are integers (or multiples of $^1/_n$ in the case when n subunit steps are used).

Grid lines. ( ) Use snapping to all horizontal and vertical lines that join two grid points. For a standard Cartesian grid, this means that we snap to points (x, y) for which x or y is an integer (or a multiple of $^1/_n$ in the case when n subunit steps are used).

Grid-curve intersections. ( ) For snapping to intersection points between curves drawn by the user and horizontal or vertical lines that join two grid points.

Curve points. ( ) Snap to arbitrary points on curves.

Curve intersections. ( ) Snap to intersections between curves, including self-intersections.

Text corners. ( ) Text objects and mathematical formulas are surrounded by an invisible rectangular bounding box. The cursor may be snapped to the corners of this box as well as to the midpoints of the edges. This kind of snapping is useful for the creation of commutative diagrams, as explained in section 4.8.

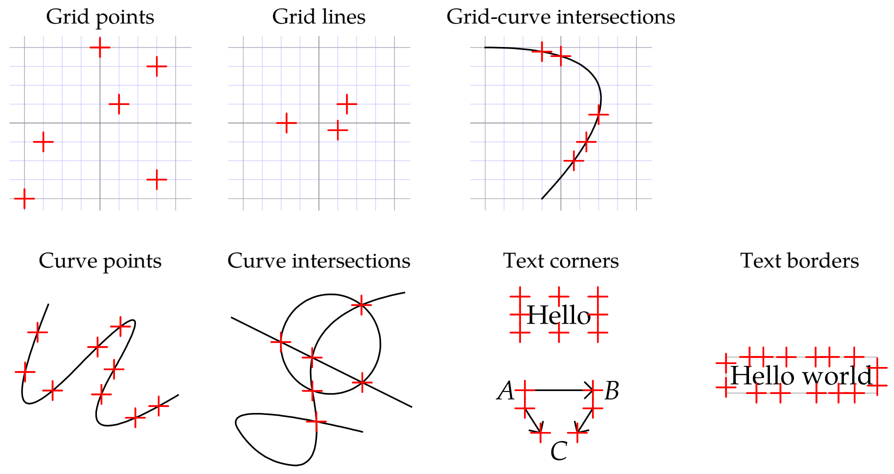


Figure 8.27. Illustration of the different kinds of snapping that are supported.

Text borders. (T+) This variant of the above way of snapping can be used for snapping to arbitrary points on the bounding box. This is useful if your text needs to be connected to many other graphical elements using arrows.

Two additional parameters control the precise way that snapping takes place. The "snapping distance" can be changed using Snap ▸ Snap distance; when approaching a special point closer than this distance, your cursor will snap. The default snapping distance is 10 pixels. People who are skilled with the mouse may prefer a smaller distance, whereas larger distances are typically more adequate on high resolution screens. When drawing pictures that contain many small details, snapping occasionally becomes annoying. Besides changing the snapping distance, another useful trick for such situations is to zoom in using +, edit the details, and then zoom back out using -.

For text objects and mathematical formulas, you may also specify the "text padding" using Snap ▸ Text padding. This quantity controls the additional padding to be inserted between the physical bounding box of the text and the logical bounding box for snapping purposes.

Interactive geometry programs such as CINDERELLA, DR. GEO, or GEOGEBRA [48, 12, 30] implement a further improvement with respect to snapping: *geometric constructions*. Assume for instance that we constructed a picture with two intersecting circles C_1 and C_2, together with a line segment ℓ between the two intersection points. In interactive geometry programs, this line segment ℓ is automatically updated when moving the control points of C_1 or C_2. In TEXMACS, the cursor is only snapped to the correct intersection points when drawing ℓ. If you change C_1 or C_2 afterwards, then you will have to move the endpoints of ℓ to the new intersection points.

Chapter 9

Laptop presentations

9.1 Introduction

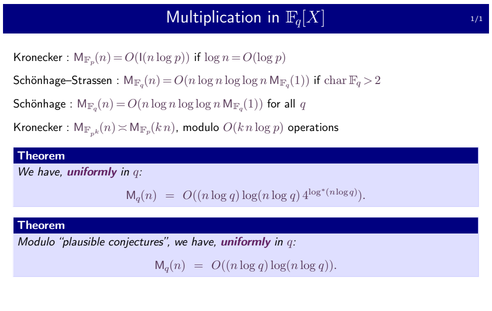
TEXMACS provides a special `beamer` style for the creation of laptop presentations. A beamer presentation consist of a series of screens instead of pages. During a presentation, the screens are displayed one after another, via a remote controller or special keyboard shortcuts. Using special markup, the screens can be further subdivided, so as to make part of their content appear progressively or in alternation. The presentation may also contain dynamic elements such as animations or on-the-fly mathematical computations.

There exist two main types of screens: textual and graphical ones. *Textual* screens are simply pages of the size of your screen and similar formatting rules apply as on ordinary paper. In particular, a consistent layout is guaranteed throughout the presentation. *Graphical* screens are pictures of the size of your screen with an optional title. Text can be placed anywhere inside such pictures and freely moved around. It is also easy to insert other graphical objects such as polygons, circles, and more general curves.

During an actual presentation, it is important to use the entire screen and remove all menu bars, toolbars, and other clutter. This is the purpose of the *presentation mode*, which can be activated and deactivated using View ▸ Presentation mode or `^F9`. The main keyboard shortcuts for traversing your slides forward and backward are `F11` and `F10`. The keys `F9` and `F12` are used to go to the start and end of the presentation, whereas `⇟` and `⇞` allow you to directly move forward and backward by one screen at a time. TEXMACS also provides a *panorama mode* which can be toggled using View ▸ Show panorama or `^F10`. This mode provides a global overview of the entire presentation and allows you to quickly move the cursor to any position.

During live performances, a remote controller is often preferred for navigating through the presentation. TEXMACS supports several types of remote controllers. Some of them (such as Apple infrared controllers) should work out of the box. Others map the buttons on the remote controller to certain keys on your keyboard, and you will need to toggle View ▸ Remote control in order to remap these keys to the right actions during presentations. If necessary, the appropriate mappings can be specified in Edit ▸ Preferences ▸ Keyboard ▸ Remote control.

Bluish



Multiplication in $\mathbb{F}_q[X]$ 1/1

Kronecker : $\mathsf{M}_{\mathbb{F}_p}(n) = O(\mathsf{I}(n \log p))$ if $\log n = O(\log p)$

Schönhage–Strassen : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ if char $\mathbb{F}_q > 2$

Schönhage : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ for all q

Kronecker : $\mathsf{M}_{\mathbb{F}_{p^k}}(n) \asymp \mathsf{M}_{\mathbb{F}_p}(k n)$, modulo $O(k n \log p)$ operations

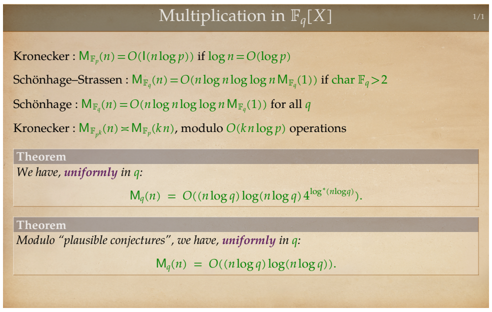
Theorem
*We have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)\, 4^{\log^*(n \log q)}).$$

Theorem
*Modulo "plausible conjectures", we have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)).$$

Parchment



Multiplication in $\mathbb{F}_q[X]$ 1/1

Kronecker : $\mathsf{M}_{\mathbb{F}_p}(n) = O(\mathsf{I}(n \log p))$ if $\log n = O(\log p)$

Schönhage–Strassen : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ if char $\mathbb{F}_q > 2$

Schönhage : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ for all q

Kronecker : $\mathsf{M}_{\mathbb{F}_{p^k}}(n) \asymp \mathsf{M}_{\mathbb{F}_p}(k n)$, modulo $O(k n \log p)$ operations

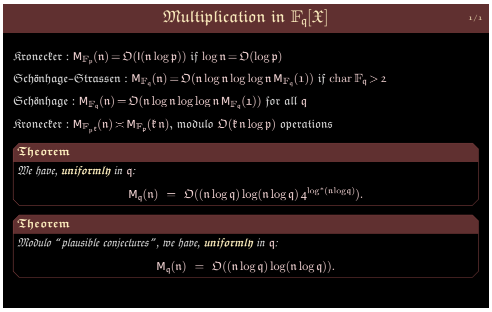
Theorem
*We have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)\, 4^{\log^*(n \log q)}).$$

Theorem
*Modulo "plausible conjectures", we have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)).$$

Dark vador



Multiplication in $\mathbb{F}_q[X]$ 1/1

Kronecker : $\mathsf{M}_{\mathbb{F}_p}(n) = O(\mathsf{I}(n \log p))$ if $\log n = O(\log p)$

Schönhage–Strassen : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ if char $\mathbb{F}_q > 2$

Schönhage : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ for all q

Kronecker : $\mathsf{M}_{\mathbb{F}_{p^k}}(n) \asymp \mathsf{M}_{\mathbb{F}_p}(k n)$, modulo $O(k n \log p)$ operations

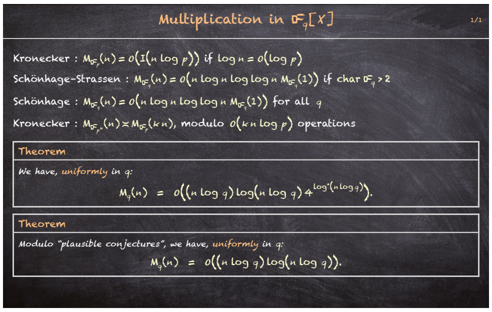
Theorem
*We have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)\, 4^{\log^*(n \log q)}).$$

Theorem
*Modulo "plausible conjectures", we have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)).$$

Blackboard



Multiplication in $\mathbb{F}_q[X]$ 1/1

Kronecker : $\mathsf{M}_{\mathbb{F}_p}(n) = O(\mathsf{I}(n \log p))$ if $\log n = O(\log p)$

Schönhage–Strassen : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ if char $\mathbb{F}_q > 2$

Schönhage : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log\log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ for all q

Kronecker : $\mathsf{M}_{\mathbb{F}_{p^k}}(n) \asymp \mathsf{M}_{\mathbb{F}_p}(k n)$, modulo $O(k n \log p)$ operations

Theorem
*We have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)\, 4^{\log^*(n \log q)}).$$

Theorem
*Modulo "plausible conjectures", we have, **uniformly** in q:*

$$\mathsf{M}_q(n) = O((n \log q) \log(n \log q)).$$

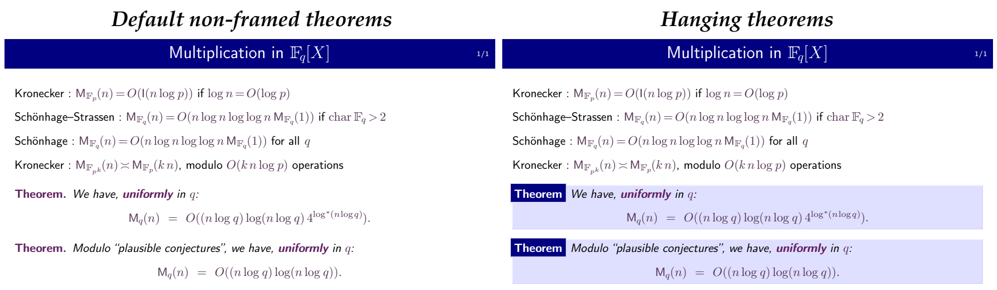
Figure 9.1. Various beamer themes.

By activating the debugging tool Tools ▸ Debugging tool and Debug ▸ keyboard, you may determine the particular mappings used by your remote control.

9.2 Themes and style options

You may select one of the beamer *themes* in order to specify the global look and feel of your presentation. Such a theme mainly corresponds to a collection of colors and background patterns for titles, mathematical formulas, theorems, etc., and the presentation itself. Assuming that the focus is on your entire presentation, the available themes can be found in Focus ▸ Beamer theme; see Figure 9.1 for a few examples.

In addition to the main theme, we recall from section 2.7 that the rendering of individual markup elements can be further customized via the Focus ▸ Preferences menu. For instance, inside a screen title, you may select Focus ▸ Preferences ▸ Framed titles as an alternative for the default title bar rendering at the top of the screen. Similarly, inside a theorem, you may select the alternative Hanging theorems rendering. Yet another example is the use of an alternative color for mathematical formulas. See Figure 9.2 for the effect of such customizations.



Default non-framed theorems

Multiplication in $\mathbb{F}_q[X]$ 1/1

Kronecker : $\mathsf{M}_{\mathbb{F}_p}(n) = O(\mathsf{I}(n \log p))$ if $\log n = O(\log p)$

Schönhage–Strassen : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log \log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ if $\operatorname{char} \mathbb{F}_q > 2$

Schönhage : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log \log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ for all q

Kronecker : $\mathsf{M}_{\mathbb{F}_{p^k}}(n) \asymp \mathsf{M}_{\mathbb{F}_p}(k\,n)$, modulo $O(k\,n \log p)$ operations

Theorem. *We have, **uniformly** in q:*

$$\mathsf{M}_q(n) \;=\; O((n \log q) \log(n \log q)\, 4^{\log^*(n \log q)}).$$

Theorem. *Modulo "plausible conjectures", we have, **uniformly** in q:*

$$\mathsf{M}_q(n) \;=\; O((n \log q) \log(n \log q)).$$

Hanging theorems

Multiplication in $\mathbb{F}_q[X]$ 1/1

Kronecker : $\mathsf{M}_{\mathbb{F}_p}(n) = O(\mathsf{I}(n \log p))$ if $\log n = O(\log p)$

Schönhage–Strassen : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log \log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ if $\operatorname{char} \mathbb{F}_q > 2$

Schönhage : $\mathsf{M}_{\mathbb{F}_q}(n) = O(n \log n \log \log n\, \mathsf{M}_{\mathbb{F}_q}(1))$ for all q

Kronecker : $\mathsf{M}_{\mathbb{F}_{p^k}}(n) \asymp \mathsf{M}_{\mathbb{F}_p}(k\,n)$, modulo $O(k\,n \log p)$ operations

Theorem *We have, **uniformly** in q:*

$$\mathsf{M}_q(n) \;=\; O((n \log q) \log(n \log q)\, 4^{\log^*(n \log q)}).$$

Theorem *Modulo "plausible conjectures", we have, **uniformly** in q:*

$$\mathsf{M}_q(n) \;=\; O((n \log q) \log(n \log q)).$$

Figure 9.2. Customized rendering for various markup elements.

The top-level focus menu also allows you to specify an aspect ratio for your presentation. Traditional projectors use a ratio of 4:3 or 5:4. When connecting your laptop to a widescreen television or certain recent projectors, you may need to select a ratio of 16:9 or 8:5 instead. In case of doubt, we recommend to opt for a 4:3 or 8:5 ratio.

When preparing a presentation, it is sometimes nice to see all screens in succession, rather than editing them one by one. This can be achieved by selecting the Paper page mode from the ☐ icon menu on the focus toolbar or using Document ▸ Format ▸ Page rendering ▸ paper. Alternatively, the Panorama mode allows you to see miniature versions of all your slides on a single screen. The paper and panorama page modes are also useful whenever you need to reorganize your presentation. Indeed, they allow you to select ranges of screens and to copy and paste them.

9.3 Switching, folding, and unrolling

The root of a presentation consists of a screens tag. This tag is an example of a *switch*: it contains an arbitrary number of screens as its arguments, but only one screen is displayed at each time. The Insert ▸ Fold ▸ Switch menu provides several other types of switches, each of which allows you to show different pieces of text in a successive and mutually exclusive manner. The most common Standard switch is used for paragraph-wide content (similar to "displayed" formulas), whereas you should use Tiny switches for inline text (similar to "inline" formulas).

Inside a switch, new *branches* can be inserted after or before the currently visible branch using Focus ▸ Insert argument after or Insert argument before; you may also use ⌥→, ⌥←, or one of the icons ⊕ or ⊕ on the focus toolbar. We also note that switches can be nested inside each other. For example, it is perfectly allowed to insert, say, a standard switch inside a screens switch. When traversing a presentation forwards using F11, you will only move to the next screen of the screens switch after traversing all branches of the innermost switch.

Another popular way to traverse a presentation is to progressively unroll content. This can be done by inserting one of the *unrolling* tags from Insert ▸ Fold ▸ Unroll:

Compressed. When using this default unrolling style, the branches that have not been unrolled are simply ignored. As a consequence, if the unroll tag is followed by other text, then this subsequent text will move downwards while unrolling.

Phantoms. When using this style, the branches that have not yet been unrolled are "displayed" using invisible ink. If the unroll tag has not completely been unrolled, then there will be blank lines between the unroll tag and subsequent text (if any).

Greyed. For this style, the branches that have not yet been unrolled are "pre-displayed" using lightly visible ink.

These primitive unrolling tags can be combined with the various kinds of item lists described in section 3.6. For instance, Insert ▸ Fold ▸ Unroll ▸ Itemize inserts a standard item list that can be unrolled progressively according to the default, compressed unrolling style. Using the standard mechanism of structured variants (see section 2.7), one may next replace the itemize and unroll tags by any of their variants.

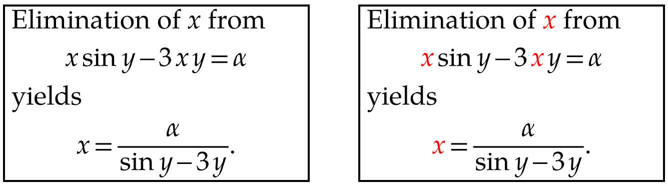
Inside a list that can be unrolled, pressing ↩ creates a new unrollable branch with a list item. Sometimes you may want to unroll more than one item at once; in that case, you have to manually insert additional items to the branch, e.g. by typing \ i t e m ↩.

A variant of unrolling is *unfolding*, in which case there are exactly two branches, one of which is always visible, and one of which can be "folded". Different folding styles are available through Insert ▸ Fold ▸ Folded. Some of the rendering styles display a button that may be pushed in order to fold or unfold. The input-output fields inside computer algebra sessions are also foldable.

Yet another variant of folded structures is available through Insert ▸ Fold ▸ Summarize. The tags in this menu are switches with two branches, again with different rendering styles.

When using TeXmacs in combination with an external plug-in, such as a computer algebra system, you will notice that all input-output fields in sessions are foldable. In addition, you can create so-called "executable switches" using the items in the Insert ▸ Fold ▸ Executable submenu. This allows you to switch back and forth between a given input and the corresponding output.

We already explained that switches may be nested in a natural way. The same holds for the other tags that we have just described. Moreover, in the Insert ▸ Fold ▸ Traversal menu, you may specify whether unrolled and unfolded structures should be folded back after traversal.



Elimination of x from

$$x\sin y-3xy=\alpha$$

yields

$$x=\frac{\alpha}{\sin y-3y}.$$

Elimination of x from

$$x\sin y-3xy=\alpha$$

yields

$$x=\frac{\alpha}{\sin y-3y}.$$

Figure 9.3. Example of highlighting a variable x when switching from one screen to a next one.

9.4 Overlays

The mechanisms of switches, unrolling, and unfolding cover the most frequent kinds of traversal of a slide show. However, there are cases in which more complex successions are needed. For example, imagine that, for a given screen, we wish to highlight all occurrences of the variable x in red on the next screen (see Figure 9.3). This could be achieved by using a switch tag: we copy the whole screen to both the first and second branch of the switch, and then color red all instances of x in the second branch. However, this solution has the disadvantage that any *a posteriori* modification on the slide has to be made both in the first and in the second branch.

TeXmacs provides the "overlay" mechanism for this kind of more complex successions of screens. You may insert a pile of overlays using Insert ▸ Fold ▸ Overlays ▸ Standard. At the start, the pile contains a unique overlay, but new overlays can then be added using the standard keyboard shortcuts ⌥→ and ⌥← for structured insertion. When applied to overlays, the standard keys F10 and F11 for traversing the presentation have the effect of moving up and down in the pile of overlays.

By default, all text that is typed by the user will be visible on all overlays. But, using the filters in the menu Insert ▸ Fold ▸ Overlay, it is also possible to create text that is only visible on specified overlays in the pile. There are four basic types of visibility filters:

Visible from here on. Text that is visible on this and all subsequent overlays.

Visible until here. Text that is visible on this and all previous overlays.

Visible only here. Text that is visible only on this overlay.

Visible except here. Text that is visible on all but the current overlay.

As in the case of unrolling, the "invisible" or "hidden" overlays can be displayed in several ways. For standard overlays (which are inserted using Standard or Compressed in Insert ▸ Fold ▸ Overlays), invisible text is simply ignored. When using Phantoms, the invisible content is rendered using invisible ink;

this corresponds to the intuitive concept of a pile of transparent layers. The last Greyed style displays the "hidden" overlays, but "greys them out" using lightly visible ink.

The precise way in which "shown" and "hidden" content are rendered for overlays is controlled by two tags shown* and hidden* that can be redefined or customized by the user. In particular, different color alternation schemes can be specified for a region of text using Insert ▸ Fold ▸ Overlay ▸ Specify color alternation. For instance, in order to achieve the effect mentioned in the example from Figure 9.3, you may specify a "black to red" color alternation, and then use a Visible from here on type of overlay.

Instead of switching between "shown" and "hidden" states depending on the overlay, it is also possible to switch between two alternate texts: depending on whether a certain predicate is met, we show the "main content" on certain overlays and the "alternate content" on the remaining ones. $\mathrm{T_EX_{MACS}}$ provides the following alternate state switches:

Alternate from here on. The alternate text will be used on this and all subsequent overlays.

Alternate until here. The alternate text will be used on this and all previous overlays.

Alternate only here. The alternate text will only be used on this overlay.

Alternate except here. The alternate text will be used on all but the current overlay.

Notice that the implementation of alternations still relies on the shown* and hidden* tags. In particular, when using the Greyed type of overlays, the "hidden" alternative will only be greyed out, and superposed with the "shown" one.

When editing overlays, the focus bar indicates the overlay that you are currently working on. It also allows you to quickly jump to any of the other overlays. Similarly, if your cursor is inside a visibility filter, then the focus bar indicates the overlays on which the tag will be in "shown" state using a bold font. You may also control the slide on which the visibility changes.

9.5 Decorations

In order to decorate your laptop presentations, $\mathrm{T_EX_{MACS}}$ provides a few extra markup elements: granite, manila-paper, metal, pine, ridged-paper, and rough-paper. These tags will put your content on a nice, natural background, as illustrated in Figure 9.4. When using the `beamer` style, these decorations are available in the Insert ▸ Prominent menu.

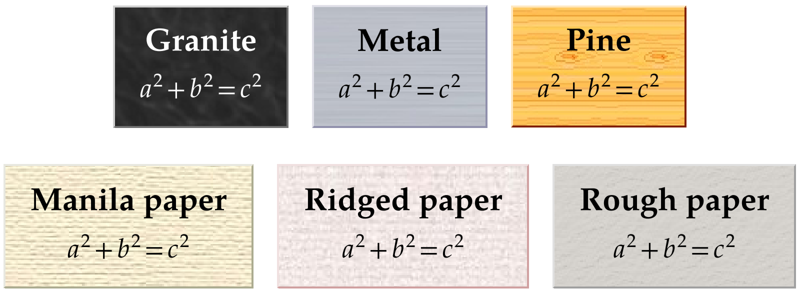


Figure 9.4. Some standard ornaments for decorating your presentations.

9.6 Graphical screens

By default, new screens are textual and untitled. It is possible to transform a newly created textual screen into a graphical one using Focus ▸ Draw, or by clicking on the Draw button on the focus bar. A title may be entered similarly, using Focus ▸ Title.

For subsequent screens, the style of the previous screen will be replicated. Assume for instance that we created a graphical screen with a title. Then newly inserted screens will automatically be graphical and titled as well. You may remove a title by selecting it using the mouse and pressing ⌫. The same holds for the picture on a graphical screen, but in this case the selection has to be performed with some care at the extremities of the picture. Indeed, clicking inside the picture makes you enter the graphics mode.

Graphical screens are essentially pictures that can be edited as explained in chapter 8. Nevertheless, for pictures in a laptop presentation, a few special rules apply. The most important changes concern the insertion of new screens and the manipulation of overlays, which are done via the menus Insert ▸ Slides, Insert ▸ Overlays, and . There is also a special mode for animating graphical objects (Insert ▸ Animate objects or); see section 9.7.3.

The Insert ▸ Slides menu is a substitute for the top-level focus menu of a slide show. It allows you to insert or remove screens without leaving the graphics mode. The first part of the menu has the same purpose.

The Insert ▸ Overlays menu can be used for the creation and manipulation of overlays in graphics mode. Graphical overlaying always applies to the entire picture. An existing picture with a single layer can be transformed into a picture with more layers using Insert ▸ Overlays ▸ Insert overlay before or Insert ▸ Overlays ▸ Insert overlay after. As soon as there are multiple layers, each graphical object admits an additional property that controls on which overlays the object should be visible. This property is shown on the toolbar after Overlay. It can be modified via the same toolbar or Focus ▸ Overlay mode, in a similar way as other properties such as the color. The currently visible overlay is also indicated on the focus toolbar and a pulldown menu allows you to quickly jump to any other overlay.

9.7 Animations

TeXmacs includes increasingly good support for animations in laptop presentations. Although a few cleverly placed animations can make your presentation slick and impress the audience, one should always be careful to not overuse this feature. Indeed, animations tend to grab all attention and thereby distract from the actual content of your talk.

You should always ask yourself how, and if, animations add something meaningful to the main discourse, rather than providing a superficial amusement. For instance, an animated arrow between two text boxes naturally directs attention from one point to another, whereas a progressively appearing text or an impressive three dimensional screen transition is usually more "for show". Nevertheless, progressively appearing text may be appropriate when unfolding output of a computation, so as to remind of the fact that computations usually need time. Even drastic animations may be appropriate at key moments of a talk: at the very start or end, when stating the main theorem, or whenever a little show can help relax the audience.

TeXmacs provides three main types of animations: basic animated effects, user-composed animations, and transformed animations. Some of the mechanisms are reminiscent of the ones described in the previous subsections for switches, overlays, etc., except that here time is continuous rather than discrete. Other atomic animations are "animated gif pictures", which can be inserted from Insert ▸ Animation ▸ Animation, and sounds, which can be inserted from Insert ▸ Animation ▸ Sound. Support for movies should be added later.

Each animation in a presentation comes with its own individual player and it starts as soon as it appears. For example when unrolling a list of animations, a new animation will appear and be played every time that you press `F11`; the animations that were already started before will just keep playing. You may use the `^F12` keystroke in order to replay all currently visible animations.

9.7.1 Basic animations

The most basic animations allow you to make some content appear or disappear gradually. TeXmacs provides various ways for doing this in the menus Insert ▸ Animation ▸ Appear and Insert ▸ Animation ▸ Vanish.

For instance, assume that you wish to let the greeting "Hello world" gradually fade in. After typing the text, it suffices to select it, and apply Insert ▸ Animation ▸ Appear ▸ Fade. On the focus toolbar, you will see a Duration field that you may use to specify the length of the animation. The duration can also be decreased or increased using the shortcuts `⌘←` and `⌘→`. We recommend durations between 0.25s and 1s. Besides fading in, TeXmacs supports gradual sliding (Translate), progressive uncovering (Progressive), and zooming in (Zoom): see Figure 9.5.

Effect	$t=0$	$t=\frac{1}{3}$	$t=\frac{2}{3}$	$t=1$
Translate		orld	o world	Hello world
Progressive		Hell	Hello w	Hello world
Fade		Hello world	Hello world	Hello world
Zoom		Hello world	Hello world	Hello world

Figure 9.5. Animations for gradual appearance of text.

Effect	$t=0$	$t=\frac{1}{3}$	$t=\frac{2}{3}$	$t=1$
Translate →		orld	o world	Hello world
Translate ←		Hell	Hello w	Hello world
Translate ↑		Hello world	Hello world	Hello world
Translate ↓		Hello world	Hello world	Hello world
Progressive ←		orld	o world	Hello world
Progressive ↑		Hello world	Hello world	Hello world
Progressive ↓		Hello world	Hello world	Hello world
Progressive ✕		o w	ello wor	Hello world

Figure 9.6. Sliding in and progressive appearance of text for various directions.

For each of the appearance and vanishing effects, the focus toolbar also contains fields with additional parameters for the animation. In the case of fading in, this allows you to specify the initial intensity. In the case of translations, you may specify the direction in which the sliding takes place, either using the → icon, or using the more detailed fields Start x and Start y. The default direction for both Translate and Progressive is from left to right. The effect of a few other directions is illustrated in Figure 9.6.

The appearance and vanishing animations are special cases of "alterations". When applying a general translation Insert ▸ Animation ▸ Alter ▸ Translate to some piece of text, you may specify arbitrary positions for both the start and the end. This may be useful if you want to insert an animation that shows the "end credits" (or references) for your presentation. In that case, you should specify 0 for both Start x and End x, take −1 for Start y, and 1 for End y. Notice that the animations described so far only work for inline content. In order to apply them to block content (i.e. text that runs over multiple lines or paragraphs), you should put your text inside a table and enable line wrapping.

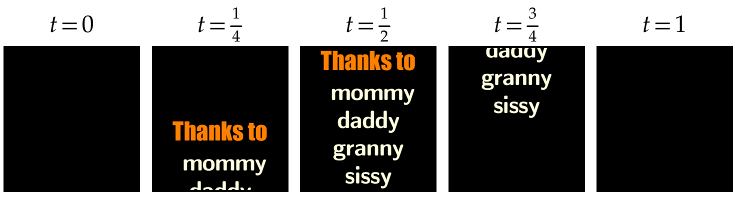


Figure 9.7. Animation for the end credits.

Effect	$t=0$	$t=\frac{1}{3}$	$t=\frac{2}{3}$	$t=1$
Shadowed	Hello world	Hello world	Hello world	Hello world
Emboss	Hello world	Hello world	Hello world	Hello world
Outlined emboss	**Hello world**	**Hello world**	**Hello world**	**Hello world**

Figure 9.8. Three dimensional animations for emphasizing text.

Further basic animations are available in Insert ▸ Animation ▸ Emphasize. The purpose of these animations is to emphasize text through an animation that catches the eye. Currently, three different kinds of three dimensional "jumping out" have been implemented: see Figure 9.8.

9.7.2 Merging and time bending

There are several ways to transform existing animations or combine them into new ones. By default, when playing two animations next to one another, they are played simultaneously. Using Insert ▸ Animation ▸ Compose, it is also possible to play several animations one after another. The individual elements of a composed animation are typically fixed animations of a given duration, which can be inserted using Insert ▸ Animation ▸ Fixed. You may also use moving or progressive content as a building block. An animation can be repeated indefinitely using Insert ▸ Animation ▸ Repeat. This may for instance be used to create blinking text.

Another type of transformation is "time bending", which amounts to accelerating certain parts of the animation while decelerating others. For the basic animations from the previous subsection, the playing rhythm can be modified via Focus ▸ Time evolution or the / menu on the focus toolbar. For general animations, time bending can be applied using Insert ▸ Animation ▸ Retime.

The most useful type of time bending consists of making the animation start and/or end more smoothly. This corresponds to what happens if you physically move an object: indeed, it always takes some time to acquire speed. It is often a good idea to end appearance animations smoothly. Another supported time transformation is "bumping", in which case the animation is played and then replayed in the reverse direction. This is often nice for animations that temporarily draw attention to some text. We finally note that the time evolution can be reversed by toggling Focus ▸ Time evolution ▸ Reverse. This turns an appearance animation into a vanishing one and *vice versa*. The types of time bending that we just discussed are illustrated in Figure 9.9.

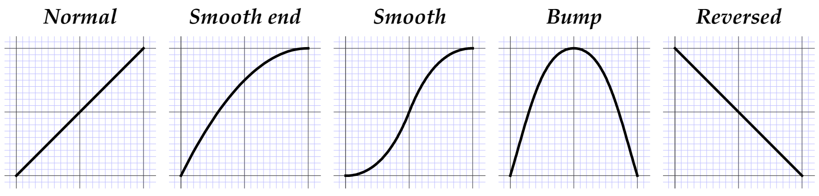


Figure 9.9. Some available time bending modes.

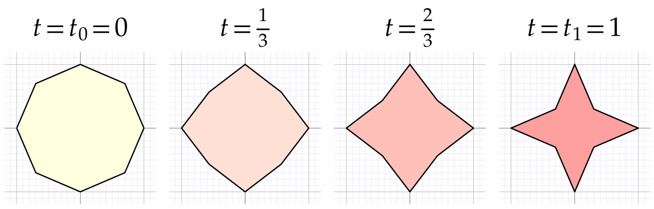


Figure 9.10. Simple example of morphing.

9.7.3 Continuous deformations

One of the most powerful mechanisms in TeXmacs for the creation of animations is *morphing*. The idea is to explicitly create a finite number of *control frames* at times $t_0 < t_1 < \cdots < t_{l-1} < t_l$. At intermediate times $t_i < t < t_{i+1}$, the content to be displayed is the result of a continuous deformation of the frame at time t_i into the frame at time t_{i+1}.

A simple example with just two control frames $t_0 = 0$ and $t_1 = 1$ is shown in Figure 9.10. Continuous deformations are only possible between similar objects. For this reason, the two control frames both contain an octagon; the only differences are the positions of the control points and the fill color.

In general, all numerical quantities, lengths, and colors can be morphed. Those parts of the control frames that differ in any other ways will usually not deform continuously. For example, TeXmacs cannot continuously deform "**Hello**" into "*Hello*", but it *can* continuously deform the slope of slanted text that was entered using Format ▸ Font effects ▸ Slanted: hello → *hello* → *hello* → *hello*. Indeed, this slope is a numerical parameter that can be specified on the focus toolbar. Other tags with numerical, length or color parameters can be animated similarly.

For animations of large pieces of text or graphics, the TeXmacs morphing algorithm attempts to automatically detect objects in the control frames that can be morphed one into another. This is useful when animating a picture in which some of the graphical objects may appear or disappear depending on the control frame. However, one should not expect any miracles: if the detection algorithm fails on a given object, then continuous morphing will be disabled for this object.

In order to animate an existing piece of text or graphics, you must select it using the mouse, and do Insert ▸ Animation ▸ Animate. You will be prompted for the duration of the animation and an animation toolbar will automatically appear with a timeline (you may also use View ▸ Animation toolbar for controlling the visibility of this toolbar). By clicking on the timeline, you may move to the corresponding control frame and edit it. As soon as you are done, press the ▶ button in order to activate your animation.

The rendering of a document fragment often depends on parameters such as a color, the position of some object, or the slope of slanted text. Nice animations can be obtained by continuously modifying such parameters. For instance, if you wish to progressively embolden some text, then you may proceed as follows:

1. Select the text and embolden it using Format ▸ Font effects ▸ Embold.
2. Set the initial emboldening strength to 0 in the focus toolbar.
3. Select the text (together with the embold tag) and do Insert ▸ Animation ▸ Animate.
4. In the time line, go to the last control frame of the animation.
5. Set the final emboldening strength to the desired value.
6. Watch your animation by pressing the ▶ button.

You may re-edit the animation using the ✎ icon on the animation toolbar. It is also possible to replay the animation as many times as you wish using the ▶ button or `^F12`.

Concerning the timeline, we note that the current frame is indicated via an asterisk $*$, whereas control frames are indicated using plus signs $+$. You may suppress a control frame using the − button, and replace the entire animation by the current frame using ✗. When animating a graphical picture, the timeline contains a few additional icons that allow you to specify when newly inserted objects should become visible or invisible. For example, assume that we are editing an animation of ten seconds and that the current frame corresponds to the frame when $t = 5\text{s}$. If you are in the default ⇢ mode, then a newly inserted curve will be visible between $t = 5\text{s}$ and $t = 10\text{s}$. In the ⇠ mode, it will rather be visible between $t = 0\text{s}$ and $t = 5\text{s}$. In the ↔ mode, the curve will be visible throughout the whole animation.

9.7.4 Graphical animations

The morphing mechanism from the previous section regards your animation as a succession of frames that are edited on a frame-by-frame basis. An alternative point of view is to regard an animation as a collection of animated objects that live their own lives and that can be edited on an object-by-object basis. Inside a graphical picture, individual objects can be animated according to this philosophy. This is most adequate if you wish to sprinkle an existing picture with a few animated elements.

In order to animate one of the objects inside a picture, you should select the animation mode using 🎞 or Insert ▸ Animate objects. It is recommended that you also activate the animation toolbar using View ▸ Animation toolbar. When clicking on a graphical object such as a line segment, a Status field appears on the focus bar. The default "inanimated" status can be changed via the popup menu that opens when you click on it. $\text{T}_\text{E}\text{X}_{\text{MACS}}$ offers the following types of animations (see Table 9.11):

Ink in. Emulate the movement of a pen that draws a curve. This is most useful for animated arrows that progressively point from a starting point to some end point.

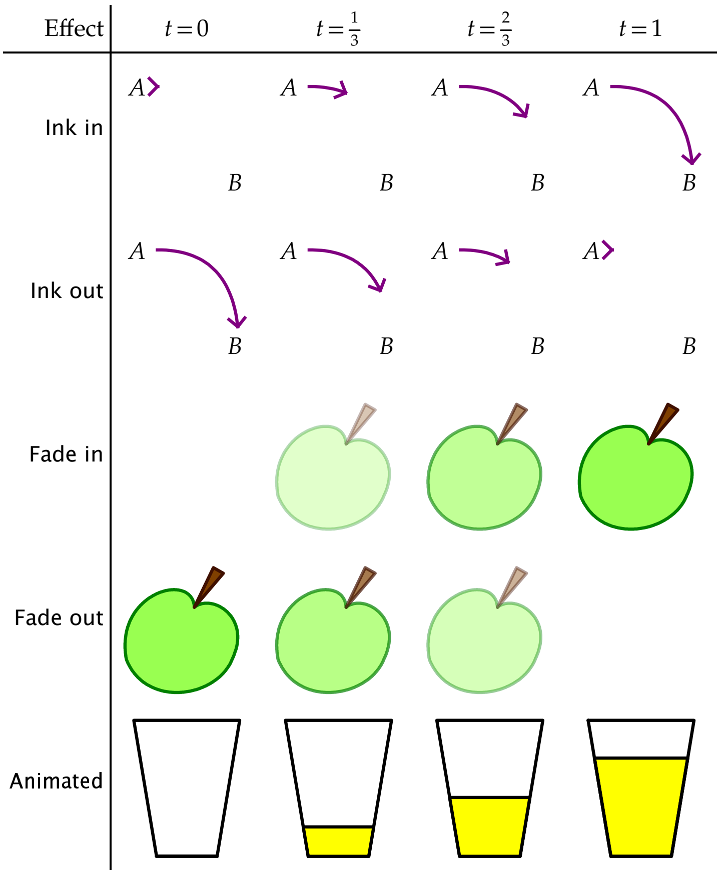


Figure 9.11. Basic graphical animations.

Ink out. The same as Ink in, but it the reverse direction. This allows you for instance to obtain a "retracting arrow".

Fade in. Smoothly increase the opacity from 0% to 100%.

Fade out. Smoothly decrease the opacity from 100% down to 0%.

For more general animations, you may use Focus ▸ Status ▸ Animated. This option works similarly to the morphing mechanism that we explained in the previous section: the animation toolbar displays a time line and you may edit the control points of your graphical object at different times. The last row in Figure 9.11 shows an example: we first drew the filled glass at $t=1$, then animated the object, and finally moved the two upper control points of the yellow liquid at $t=0$ so as to coincide with the two lower control points.

Graphical animations of the type that we just described have the advantage that they behave similarly to ordinary graphical objects in many respects. For example, it is easy to move an entire animation to a different place in the picture or to copy and paste it. On the other hand, the control points can only be edited in the individual time frames attached to the animation.

9.8 Exporting beamer presentations

At certain venues, it is not possible to use your own laptop for making presentations. In such cases the organizers often impose a specific file format for presentations such as PDF or POWERPOINT. TeXmacs has native support for exporting a beamer presentation as a PDF file. This works well for basic presentations, but there are several quirks if you make use of all functionality provided by TeXmacs:

- Although the PDF reference format specifies mechanisms for the inclusion of animations and 3D graphics, most PDF viewers do not implement these features. For this reason, TeXmacs animations are not exported to PDF.
- Various other "dynamic" types of markup are not exported to PDF. For instance, you may run an external computer algebra system inside a TeXmacs presentation, but only the final output can be exported to PDF.

When exporting a presentation, TeXmacs emulates the traversal of the document using Dynamic ▸ Next and generates one page for every next state. In particular, if your presentation contains folded or unrollable elements, then your PDF typically contains far more pages than the number of slides in your original presentation.

Note that the PDF export facility also provides a user preference Edit ▸ Preferences ▸ Convert ▸ Pdf ▸ Expand beamer slides. This technical option is somewhat tricky to explain, but if you are not happy with the PDF export of a presentation, then it may be worth it to try this setting and check whether it improves things for you.

At present, beamer presentations cannot be exported to LaTeX. Fortunately, such conversions are rarely needed: you usually do not send your presentations to publishing companies that tend to bother authors with specific document formats and presentation styles.

Chapter 10
Special editing facilities

In this chapter, we discuss some of the general editing facilities that are implemented in $\text{T}_\text{E}\text{X}_{\text{MACS}}$. Of course, this includes basic operations that can also be found in other editors: "cut and paste", "search and replace", etc. But, more interestingly, some of these facilities take advantage of the additional structure of $\text{T}_\text{E}\text{X}_{\text{MACS}}$ documents. Typical examples of *structured editing features* are "structured cursor movement" and "structured variants". Traditional operations such as "search and replace" can also exploit the document structure. For instance, when searching x in math mode, you will only find matches that are also in math mode.

10.1 Cut and paste

You can select text and formulas by moving the mouse while holding the left button. In order to delete the selected region, use Edit ▸ Cut or `^w`; to copy it, use Edit ▸ Copy or `⌘w`. Next, paste it as many times as you want using Edit ▸ Paste or `^y`. Alternatively, you may copy a selected region using the middle mouse button.

It is also possible to change the text properties of a selected region. For instance, in order to make some black text red, you select it using the left mouse button and select the desired red color 🟥 in Format ▸ Color. Similarly, if you select a formula and click on Insert ▸ Fraction, then the formula becomes the numerator of the new fraction.

When using the copy and paste mechanism to communicate with another application, $\text{T}_\text{E}\text{X}_{\text{MACS}}$ and the application first have to agree on a data format. $\text{T}_\text{E}\text{X}_{\text{MACS}}$ accepts multiple formats for pasted text; by default, it uses its own format for copying. Copying and pasting data in other formats is possible using Edit ▸ Copy to, Cut to, and Paste from. For instance, a LaTeX formula can be pasted inside a $\text{T}_\text{E}\text{X}_{\text{MACS}}$ formula using Edit ▸ Paste from ▸ LaTeX. You may also specify your preferred default import and export formats using Tools ▸ Miscellaneous ▸ Import selections as and Export selections as.

By default, copying and pasting uses the "primary clipboard". Using the Edit ▸ Copy to, Cut to, and Paste from menus, you may specify as many other clipboards as you like. This allows you to keep multiple selections in memory, ready to be pasted.

There are two ways to make selections using the keyboard. The first way is to use the cursor keys ←, →, etc. while simultaneously holding down the shift key ⇧. Alternatively, you may press ^␣ once to set a starting position. When moving around using the cursor keys, the text between the starting position and the current position keeps being selected. The selection gets cleared by pressing ^g.

Notice that the ^␣ shortcut also allows you to make *structured selections*. You may select the current word by pressing ^␣ twice. Each additional time you press ^␣ results in the selection of the smallest structure that encompasses the current selection. Ultimately, when the entire document gets selected, pressing ^␣ once more clears the selection.

10.2 Search and replace

Textual search and replace

You can start searching text by pressing ^s or Edit ▸ Search. Doing this, a new special "search toolbar" will appear below the main text, just above the footer. When typing text in the search field of the toolbar, all occurrences of this text will be highlighted in the main document. Moreover, one "principal" occurrence will be highlighted in red and you may navigate through all occurrences using ⇞ and ⇟ (or ↑ and ↓, or ↩). Using ↖ and ↘, you may jump to the first and last occurrences. As soon as you press the escape key ⎋, the search toolbar will be closed, searching stops, and focus returns to the main document.

During a search, TeXmacs only looks for text in the same mode and language as at the position where you started your search. In other words, when searching for x in math-mode, you will not find any x's in ordinary text (which are usually irrelevant). As a current limitation, the search string on the search toolbar can only contain ordinary text and no math-symbols or structured text. Below, we will discuss a mechanism that allows you to search for more general mathematical symbols and formulas.

In order to replace text, you should use Edit ▸ Replace or press ^=. This will cause a special "replace toolbar" to appear below the main text, just above the footer. The search field of this bar contains the string to search for, whereas the replace field contains the string to replace with. Again, you may use the ⇞ and ⇟ keys in order to navigate through the occurrences of the search string. When pressing ⇥ or ↩ in the search field, focus will be moved to the replace field. You may still use the ⇞ and ⇟ keys in order to navigate through the occurrences of the search string. In addition, pressing ↩ will replace the principal occurrence of the search string by the replace string. Using ⇧↩, you may undo the last replacement. You may replace all remaining occurrences by pressing ^↩. As in searching, the query-replace command is mode and language sensitive.

General search and replace

The search and replace toolbars are quite rudimentary in the sense that they only allow for searching and replacing plain text. By pressing the icon on either of these toolbars, you may expand the toolbar into a full-blown window with larger search and replace fields that may contain arbitrary markup. Searching and replacing can be done using more or less the same keyboard shortcuts as with the toolbars, but you may now search and replace arbitrary content. In order to switch back to the toolbar methods for searching and replacing, you have to click on the icon.

When searching non-textual content, the conditions for a match are somewhat relaxed. For example, assume that you are just starting a new search with an empty search field. Then typing F6 inserts the strong tag with no text inside yet. Instead of looking only for "strong empty strings", TeXmacs will look for all strong markup in your document. If you next enter the letter "a", then TeXmacs will look for all strong text that contains the letter "a". In a similar way, when searching for the formula $\frac{x}{}$, TeXmacs will highlight all fractions in which the numerator contains the variable x. Yet another example: a search for $\frac{}{\sqrt{x}}$ will highlight all formulas in which the denominator contains a square root that contains the variable x. For instance, the fraction $\frac{a+b}{c+\sqrt{x+y}}$ will be highlighted, but not $\frac{\sqrt{x+y}}{a+\sqrt{y}}$.

When using the structured search and replace widgets, TeXmacs also implements a few additional special tags for enhancing structured searching. First of all, it can happen that you need to search for certain content *inside* a special context. For example, you might want to search for all occurrences of the letter "a" inside a strong tag. When searching for **a**, as above, TeXmacs will highlight all strong tags that contain the letter "a". In order to highlight the letters "a" themselves, you should first insert the strong tag inside an empty search field using F6. You next type ^? to insert a special select-region tag, and finally enter the letter "a" inside this tag.

10.3 Spell checking

If the `ispell` program or any compatible substitute has been installed on your system, then you may use it to check your text for misspelled words by pressing ⌘$ or Edit ▸ Spell. Of course, you will also need the dictionaries corresponding to the languages in which your texts are written; `ispell` usually comes with an English dictionary.

The interface for spell checking is similar to the one for searching and replacing text. Misspelled words are highlighted in the same way as search hits. You may again navigate through the highlighted words using ⇞ and ⇟ (or ↑ and ↓). There is also a spell bar just above the status bar to assist you with the correction process.

More precisely, given a highlighted incorrect word, the spell bar proposes a list of possible corrections. You may select one of these corrections by typing the corresponding number or by pressing the corresponding button on the spell bar. In the absence of a suitable correction, you may explicitly enter one inside the Correct field on the spell bar, followed by ↩.

It may also happen that a correct word was not in the dictionary. In that case you may add it to the dictionary by typing + or by pushing + on the spell bar. If you are not sure, but temporarily wish to accept the word, then you may press ⇥ or →.

If your document contains text in other languages, don't forget to correctly specify the language for every piece of foreign text using Insert ▸ Language. In that case, TeXmacs will automatically use the corresponding dictionaries when performing a spell check.

10.4 Undo and redo

It is possible to gradually undo the changes you made in a document from the moment that you launched TeXmacs. This can be done via Edit ▸ Undo or using the keystroke ⌘[. Undone changes can be "redone" using Edit ▸ Redo or ⌘].

TeXmacs maintains the entire "history tree" of all your edits. This history is not necessarily linear; multiple branches in the history tree may arise as follows: type "a", undo the insertion of "a", type "b", and then again undo the insertion of "b". At this point, there are two redo options: the reinsertion of "a" and the reinsertion of "b". In such cases, Edit ▸ Redo becomes a menu in which you can select the appropriate branch.

The undo system takes into account changes in the document itself only. In particular, modifications of most of the global document properties can not be undone. This includes modifications of the document style, the page size, the main font, etc. The same remark applies to any modifications outside TeXmacs that were triggered by your actions. For instance, in a computer algebra session, you can undo your edits inside TeXmacs, but not the computations in the external computer algebra system.

10.5 Structured editing

The behavior of most structured editing operations is conditioned by the *current focus*. By default, the focus is on the innermost tag that contains the cursor. Whenever some selection is active, the focus is rather on the innermost tag that contains the selection. Occasionally, during certain structured editing operations, the focus may be determined in an *ad hoc* manner. The current focus is visually indicated through a cyan box around the cursor.

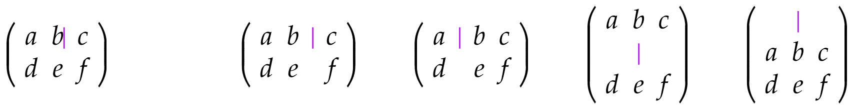


Figure 10.1. Assume that the cursor is at | inside the left-most matrix. Then the four other matrices respectively correspond to the insertion of a new column at the right (⌥→) or left (⌥←), or a new row below (⌥↓) or above (⌥↑).

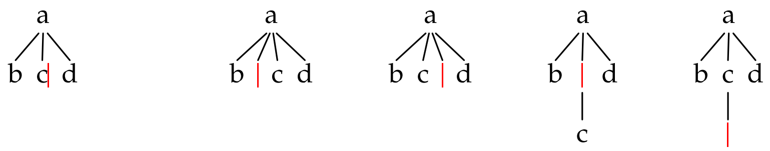


Figure 10.2. Assume that the cursor is at | inside the left-most tree. Then the four other trees respectively correspond to the insertion of a new node at the left (⌥←), at the right (⌥→), above (⌥↑) or below (⌥↓).

As a first example, consider the *structured insertion* commands ⌥←, ⌥→, ⌥↑, and ⌥↓. These operations are applicable inside both tables and trees, but with slightly different semantics. Inside tables, they allow you to insert new rows and columns (see figure 10.1). Inside trees, they correspond to the insertion of new nodes (see figure 10.2). If you are inside a tree inside a table, then the innermost tag is a tree, so the current focus will be on the tree. Consequently, node insertions take precedence over the insertion of new rows and columns.

Similarly, still in the case of tables, the keys ⌥↖, ⌥↘, ⌥⇞, and ⌥⇟ can be used for inserting a new first or last column, or a new first or last row. The keys ⌥⌫ and ⌥⌦ are mapped to the commands for backward and forward *structured deletion*. For tables, this will result in the removal of the column before or after the cursor (see figure 10.3). In order to remove the enclosing environment you may use ^⌫.

10.6 Structured cursor movement

TeXmacs implements three main types of *structured cursor movement*:

1. Traversal of the entire structure of the document.

2. Traversal of tags that are similar to the innermost tag.

3. Movements inside the innermost tag.

Most keyboard shortcuts for structured cursor movements can also be used in combination with the ⇧-key so as to simultaneously select text while moving around.

$$\begin{pmatrix} a & b| & c \\ d & e & f \end{pmatrix} \qquad \begin{pmatrix} b| & c \\ e & f \end{pmatrix} \quad \begin{pmatrix} a & |c \\ d & f \end{pmatrix} \quad b|$$

Figure 10.3. Assume that the cursor is at | inside the left-most matrix. Then pressing the keys ⌥⌫ and ⌥⌦ respectively result in the next two matrices. Pressing ^⌫ replaces the matrix by the content of the cell with the cursor, leaving you with the b at the right-hand side.

Structured traversal of the document

The ^←, ^→, ^↑, and ^↓ shortcuts are used for the structured traversal of the entire document. Inside plain text, ^← and ^→ allow you to move in a word-by-word manner, whereas ^↑ and ^↓ correspond to paragraph-by-paragraph motion.

In the presence of other markup, the ^← and ^→ shortcuts allow you to visit all accessible cursor positions in the document, except that we keep moving in a word-by-word manner inside plain text. The behavior of the ^↑ and ^↓ shortcuts is more context-dependent. Inside matrices, they typically allow you to move one row up or down.

Traversal of tags that are similar to the innermost tag

This type of cursor movement allows you to quickly visit all other tags in the document that are *similar* to the innermost tag. The ^⇞ and ^⇟ shortcuts allow you move to the previous or next similar tags, whereas ^↖ and ^↘ directly jump to the first or last similar tags.

For example, if you are inside a section title, then you may move to the previous sectional title using ^⇞. Subsection and chapter titles are in particular understood to be "similar" to section titles. Notice that you may use ^§ to jump to the title of the section you are editing.

Movements inside the innermost tag

It is also possible to quickly move inside the innermost tag without quitting it. The shortcuts ⌘⌥←, ⌘⌥→, ⌘⌥↖, and ⌘⌥↘ allow you to move to the previous, next, first, or last argument of the innermost tag. Furthermore, the shortcuts ⌘⌥⌫ and ⌘⌥⌦ may be used to exit the innermost tag on the left or on the right.

The behavior may be different in special contexts. For instance, inside tables or trees, the above shortcuts rather correspond to cell-by-cell or node-by-node cursor movement. In addition, these cases associate vertical cursor movements to ⌘⌥↑, ⌘⌥↓, ⌘⌥⇞, and ⌘⌥⇟.

10.7 Structured variants

When creating an environment like a theorem, an equation, or a list, it often happens that you need to change the environment *a posteriori*. The keyboard shortcuts ^⇥ and ^⇧⇥ allow you to cycle through the list of *structured variants* of the innermost tag, in forward or backward direction, respectively.

For instance, assuming that you are inside a theorem, pressing ^⇥ several times will change the theorem into a proposition, a lemma, a corollary, a conjecture, and finally back into a theorem. The ^⇧⇥ key allows you to cycle in the reverse direction: theorem → conjecture → corollary → lemma → proposition → theorem. Note that you may also select the desired variant in Focus ▸ Theorem or using the Theorem menu on the focus bar.

In the case of mathematical formulas, the ^⇥ shortcut allows you to change an inline formula such as $a^2+b^2=c^2$ into the displayed formula

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

The editor takes care of potential "trailing spaces and punctuation signs".

TeXmacs also provides the ^# shortcut for turning numbered environments into unnumbered ones and *vice versa* (which is equivalent to the Ⅳ icon on the focus bar). This works for most common environments like theorems, remarks, tables, equations, etc. Notice that ^# also turns an unnumbered itemize environment into an enumeration and *vice versa*, whereas ^⇥ allows you to cycle between the available kinds of list items (bullets, dashes, arrows, etc.).

Folding and unfolding provides yet another interesting example of toggling between several environments. Inside a computer algebra session such as

```
Pari] factor (x^15 - 1)
```

$$\%1 = \begin{pmatrix} x-1 & 1 \\ x^2+x+1 & 1 \\ x^4+x^3+x^2+x+1 & 1 \\ x^8-x^7+x^5-x^4+x^3-x+1 & 1 \end{pmatrix}$$

you may click on "`Pari]`" in order to fold the output (i.e. only the input remains visible) and click once again in order to unfold back to the original state. The ^* shortcut achieves the same effect. There are various other foldable environments, most of which are available through Insert ▸ Fold.

10.8 Positioning and resizing objects

The prefix ⌘ may be used for positioning and resizing objects. For instance, inside a cell of a table, you may use ⌘→ to align the cell more to the right (so a left-aligned cell becomes centered, whereas a centered cell becomes right-aligned). Behind a space introduced via Format ▸ Space, the same shortcut allows you to increase the width of space. More generally, the following shortcuts are implemented:

⌘←. Decrease the horizontal size of an object, or align more to the left.

⌘→. Increase the horizontal size of an object, or align more to the right.

⌘↓. Decrease/increase the vertical size of an object, or align farther down.

⌘↑. Increase/decrease the vertical size of an object, or align farther up.

⌘↖. Decrease the horizontal offset of an object, or left align.

⌘↘. Increase the horizontal offset of an object, or right align.

⌘⇟. Decrease the vertical offset of an object, or align at the bottom.

⌘⇞. Increase the vertical offset of an object, or align at the top.

⌘⌫. Revert the geometry (size, position, alignment) to the defaults.

⌘^⇥, ⌘^⇧⇥. Cycle among the available length units for specifying the geometry.

⌘^[, ⌘^]. Decrease or increase the step size when positioning or resizing.

These shortcuts apply in particular to the following tags:

Spaces. Both horizontal and vertical spaces from the Format ▸ Space menu. You should put the cursor just after the space tag for the shortcuts to apply.

Box modifiers. The tags move, shift, resize, extend, clipped, smash, inflate from the Format ▸ Adjust menu.

Animations. The durations can be modified using ⌘← and ⌘→.

Images. The size and alignment of images can be changed.

10.9 Versioning tools

When writing documents in collaboration with others, one frequently needs to review changes made by coauthors and either accept, discard, or further correct them. The integrated versioning tool has been designed to facilitate this task. After enabling it through Edit ▸ Preferences ▸ Utilities ▸ Versioning tool, a special Version menu appears in the main menu bar.

Independently from TeXmacs, there exist many programs for "version control", such as SUBVERSION [6, 7], GIT [62, 51], DARCS [49], GNU BAZAAR [45], etc. These systems are complementary to the built-in versioning system; they operate on the level of the file system and in particular allow you to maintain several versions of a document. TeXmacs currently provides a rudimentary support for SUBVERSION, but interfaces for other versioning systems could easily be added.

Comparing two versions

Assume that we have two versions `old.tm` and `new.tm` of the same document. In order to see the changes, first load the newer version `new.tm`, then click on Version ▸ Compare ▸ With older version, and select the older version `old.tm`. The buffer will still be named `new.tm`, and the changes between both versions will be indicated by special markup. If there are any changes, then the cursor will be positioned at the first difference. Similarly, you may compare the current buffer with a newer version on disk using Version ▸ Compare ▸ With newer version.

It is possible to go through all the differences between the old and new versions either from the items in the submenu Version ▸ Move, or using the keyboard shortcuts ^↑ and ^↓. You may also use the general structured navigation shortcuts ^↖, ^↘, ^⇞, and ^⇟.

Visualization of the differences

Differences between the two versions can be displayed in three ways: by showing only the old version, only the new version, or both versions simultaneously. In all cases, the old version is displayed in dark red and the new version in dark green: see Figure 10.4.

The visualization style can be specified individually for each individual change, via Version ▸ Show or the keyboard shortcuts ^← (old version), ^→ (new version) and ^| (both versions). One may also cycle through the different styles using the structured variant key ^⇥. If you selected some text, then the above actions will apply to the whole selection. In particular, by selecting the entire file, you can visualize the older or the newer version, or both versions.

Retaining a specific version

When going through the changes between two versions, it is possible to retain either one or the other version for each individual difference. Assuming that the cursor is inside a given difference, this can be done via the submenu Version ▸ Retain. Alternatively, you may use the shortcuts ^1, ^2, and ^↩ to retain the old, new, and currently displayed version, respectively. If both versions are displayed, then ^↩ retains the new version. After retaining one of the versions, we automatically jump to the next difference, which can then be processed.

If you selected some text, then any of the above actions will retain the appropriate version for each of the differences in the selection. This applies in particular to the case when you select the entire document. A more conservative approach to process all differences between two versions is to first review them with ^↑ and ^↓ while selecting your preferred versions with ^← and ^→. Only at the very end, once you are sure about your choices, you select the entire document and click on Version ▸ Retain ▸ Current version.

The old version of a text text with some words.	Old version
The new version of a text with some extra words.	New version
The oldnew version of a text text with some extra words.	Both versions

Figure 10.4. Three ways to display the differences between two versions.

Detailed grain	**Block grain**
Modern life	*Modern life*
When woringworking in haisthaste, it is likely that your text will be full of typos. Odds are also high that more serious mathematical mistakes will ruin your efforts.	When woring in haist, it is likely that your text will be ful of typos. Odds are also high that more serious mathematical mistakes will ruin your efforts.
	When working in haste, it is likely that your text will be full of typos. Odds are also high that more serious mathematical mistakes will ruin your efforts.
$a_1^2+\cdots+a_n^2 n\,(a_1^2+\cdots+a_n^2) \geqslant (a_1+\cdots+a_n)^2$	$a_1^2+\cdots+a_n^2 \geqslant (a_1+\cdots a_n)^2$
	$n\,(a_1^2+\cdots+a_n^2) \geqslant (a_1+\cdots+a_n)^2$

Figure 10.5. Alternative grains for visualizing differences.

Grain control and recomputing the differences

The entries in the submenu Version ▸ Grain allow you to control the grain at which differences between versions are computed. By default, we use the finest grain Detailed. It is also possible to compute differences on a paragraph-based level, using Block. In that case, the entire paragraphs in which a change occurs are highlighted, as illustrated in Figure 10.5. The roughest grain Rough will highlight the entire text, if a change occurs somewhere inside.

The grain is used when comparing two documents using Version ▸ File ▸ Compare, but it is also possible to change the grain for a selected portion of text. In particular, this can be applied on the entire buffer. Similarly, if you change the grain inside a difference, then the difference will be recomputed using the new grain.

Note that you may also "change" the grain to the current grain. This has the effect of updating the differences inside a selection or the current difference at the cursor position. This may be useful if you changed one of the two versions. For instance, assume that the old version contained a theorem and that the theorem was changed into a lemma in the new version while also modifying parts of its body. When visualizing the changes, the whole theorem will be highlighted, since there is no appropriate markup to indicate the mere change from a theorem into a lemma. Therefore, if we want to compare the bodies, the solution is to first turn the old theorem into a lemma, and then to update the differences.

Using external programs such as SUBVERSION *for version control*

If the file you are editing belongs to a directory that is under version control (only SUBVERSION is currently supported, although other systems might follow), then the Version menu contains a few extra entries.

First of all, if the current buffer is under version control, then you may take a look at its history using Version ▸ History. The history contains a list of hyperlinks to older revisions, together with short information about who changed what and when. Older revisions cannot be saved, but you may compare them with the current user version (on disk or being edited) using Version ▸ Compare ▸ With current user version.

After making some changes to a file under version control, the version inside the editor or on disk no longer corresponds to the version in the repository. Using Version ▸ Commit, the current version can be committed to the repository. When doing so, you will be prompted for a small explanatory message about the changes that you have made. A file that is not yet under version control can be added to the version control system using Version ▸ Register. Registering a file does *not* commit it to the repository: you still have to use Version ▸ Commit in order to do so.

If the repository version of the file was modified while you were editing the local one, then you may merge both versions using Version ▸ Update. No mechanisms for conflict resolution have been implemented yet, although this is planned for the future.

Chapter 11
Scientific computations

A major feature of TeXmacs is the capability to delegate mathematical computations to external *plug-ins* and present the results inside the editor. For computer algebra systems and other scientific software, this is typically done in shell-like sessions: you simply type the commands to be evaluated, after which TeXmacs presents the results in a nice, graphical way. Some systems can also be used in alternative ways, e.g. in order to evaluate a selected expression inside a formula, or as the computational engine of a spreadsheet.

One special plug-in that is always supported is SCHEME. This purified variant of LISP is one of the standard *extension languages* for the GNU project. Many aspects of the TeXmacs user interface (such as the menus, the keyboard shortcuts, etc.) can be extended or customized using the SCHEME programming language (see chapter 14). SCHEME sessions can in particular be used to test such customizations on-the-fly.

Another plug-in that is usually present is BASH, the "Bourne-Again Shell". This allows you to evaluate operating system commands from within the editor. The availability of additional plug-ins depends on the software that is installed on your computer. TeXmacs automatically activates a plug-in whenever the corresponding external program is detected on your computer. See Help ▸ Plug-ins for the list of existing plug-ins and specific documentation on each of them.

11.1 Shell-like sessions

11.1.1 Starting and ending sessions

A new *session* can be started from the Insert ▸ Session menu. As explained above, it is always possible to start a SCHEME session using Insert ▸ Session ▸ Scheme; BASH sessions are usually also available via Insert ▸ Session ▸ Shell. The further items in the Insert ▸ Session menu depend on the plug-ins that are installed on your system.

A session consists of a sequence of input and output fields and optional text between them. When pressing ↵ inside an input field of a session, the input text is evaluated and the result is displayed in an output field. A simple example of a SCHEME session goes as follows:

```
Scheme] (+ 1 1)
2
```

Most plug-ins rely on an external application for computations. When you start a new session for the first time, TeXmacs automatically launches this external application in the background. The process that is launched in this way will be shared by all subsequent sessions for the same plug-in: when "starting" another session, you are actually continuing the previous session at another place in your documents. In particular, if you launched a long computation, then you will have to wait until it completes.

Some plug-ins allow you to interrupt long computations using Focus ▸ Interrupt execution, or using the icon on the focus toolbar (assuming that your cursor is inside the session). This makes it possible to proceed with other computations in the current environment of your session. Unfortunately, "soft interruptions" of this kind have been implemented in very few plug-ins only, for the time being.

You may always force the external application to quit using Focus ▸ Close session or , at the expense of losing your computational environment. When pressing ↩ in the input of a system that is not, or no longer running, the system will be restarted automatically. In order to re-evaluate all fields of an existing session, you may use Focus ▸ Evaluate ▸ Evaluate all. Similarly, Evaluate above and Evaluate below allow you to evaluate all fields above or below the current field. Whenever a long computation had to be interrupted using Focus ▸ Close session, this may be used to restore the same working environment as just before the problematic command.

Let us emphasize that the execution of commands via external plug-ins is done in a completely asynchronous manner. This allows you to continue typing while a long computation is in progress. It also means that sessions of different plug-ins can be run in parallel. In fact, the same mechanism can be exploited to launch multiple instances of the same plug-in. More precisely, when inserting a session using Insert ▸ Session ▸ Other, you may specify both a "session type" (Shell, Pari, Maxima, etc.) and a "session name" (the default name is "default"). Only sessions with the same name and type will share the same physical process; by varying names, it thus becomes possible to simultaneously run multiple instances of the same plug-in.

We finally note that different rules apply in the case of the SCHEME plug-in. Indeed, since SCHEME is the extension language of $\text{T}_\text{E}\text{X}_\text{MACS}$, the SCHEME plug-in is permanently running: there is no need to start it and no way to stop or interrupt it.

11.1.2 Editing sessions

Inside input fields of sessions, the cursor keys have a special meaning: when moving up or down, you will move to previous or subsequent input fields. When moving to the left or to the right, you will never leave the input field (you may still do this with the mouse).

Some utilities for editing input, output, and text fields are available in the Focus menu. New fields can be inserted using ⌥↑ (insert above) and ⌥↓ (insert below). Keyboard shortcuts for removing matching text/input/output fields are ⌥⌫ (remove backwards) and ⌥⌦ (remove current field).

Example 11.1. MAXIMA is one of the oldest computer algebra systems. Nowadays, the system is freely available, while being actively maintained. A typical session is shown below. If MAXIMA is present on your system, then you may put your cursor in one of the inputs, perform some edits, and try to re-execute it. The welcome message before the first input is only displayed when a new session is started.

```
Maxima 5.25.1 http://maxima.sourceforge.net
using Lisp SBCL 1.0.51
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1) diff (x^x^x, x)
```

(%o1) $x^{x^x}(x^x \log(x)(\log(x)+1)+x^{x-1})$

```
(%i2) integrate (%o1, x)
```

(%o2) $\mathrm{e}^{\mathrm{e}^{x\log(x)}\log(x)}$

```
(%i3) integrate (x^5 / (x^2 - x + 17), x)
```

(%o3) $\frac{239\log(x^2-x+17)}{2}+\frac{1361\arctan\left(\frac{2x-1}{\sqrt{67}}\right)}{\sqrt{67}}+\frac{3x^4+4x^3-96x^2-396x}{12}$

It should be noticed that all input/output fields in sessions are *foldable*: when clicking on an input prompt such as (%i1) with the mouse, you may fold or unfold the entry. The output fields of folded entries are hidden, even though they still exist in the document.

During laptop presentations, a similar folding and unfolding process is performed automatically when traversing your slide show. It is also possible to fold or unfold all fields in a session using Focus ▸ Session ▸ Fold all fields and Unfold all fields. You may use Clear all fields to physically remove all output fields; this can be useful for creating a demo session that will be re-executed from scratch during a presentation.

For the creation of long software demonstrations, it is often convenient to group input/output fields together into *subsessions* that are labeled by theme (see Figures 11.1 and 11.2 below). A subsession is created using Focus ▸ Session ▸ Create subsession or ⌥→. In that case, the current input-output field becomes the body of the subsession. A subsession consists of explanatory text followed by the main body. Subsessions can be folded and unfolded using the mouse or ^*.

Groups of input/output fields can be cut, copied, and pasted in the usual way. You may also use Focus ▸ Session ▸ Split session in order to split a big session into smaller parts. This may prove handy whenever you wish to intersperse your session with explanatory text.

11.1.3 Input and output options

By default, TEXMACS evaluates input fields when pressing ↩. Multiline input can be created using ⇧↩. Alternatively, when selecting the multiline input mode using Focus ▸ Input mode ▸ Multiline input, the ↩ key will insert a new line and ⇧↩ may be used in order to evaluate the input field. Note that certain plug-ins come with built-in heuristics for testing whether the input has been completed; if not, then ↩ will insert a new line.

Certain plug-ins allow you to type the mathematical input in a graphical, two dimensional form. If so, this input method can be enabled using Focus ▸ Input mode ▸ Mathematical input; it is usually also possible to copy and paste output back into the input. However, this may not work perfectly for all plug-ins. Also bear in mind that the mathematical input mode may preempt key combinations that are also needed to type certain commands. For instance, the key $ is redefined inside math mode, so you have to type ⇧F5 $ in order to enter the dollar symbol.

Example 11.2. The session from example 11.1 becomes as follows when using mathematical input (Focus ▸ Input options ▸ Mathematical input):

(%i1) $\mathrm{diff}(x^{x^x}, x)$

(%o1) $x^{x^x}(x^x \log(x)(\log(x)+1)+x^{x-1})$

(%i2) $\int \%o1 \, \mathrm{d}x$

(%o2) $\mathrm{e}^{\mathrm{e}^{x \log(x)} \log(x)}$

(%i3) $\int \frac{x^5}{x^2-x+17} \mathrm{d}x$

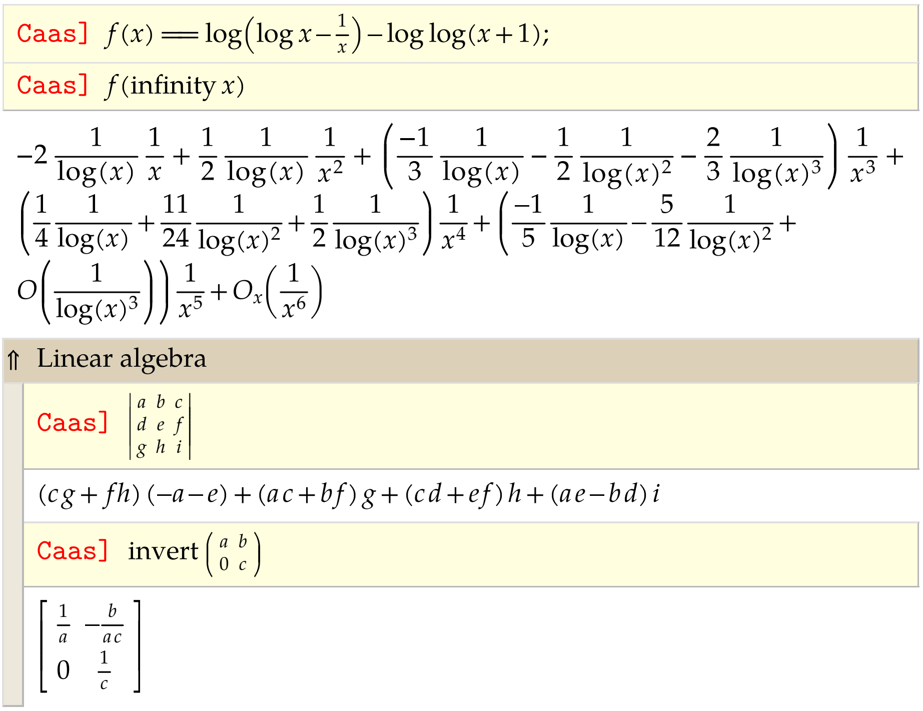
(%o3) $\frac{239 \log(x^2-x+17)}{2} + \frac{1361 \arctan\left(\frac{2x-1}{\sqrt{67}}\right)}{\sqrt{67}} + \frac{3x^4+4x^3-96x^2-396x}{12}$

There are also a few output options that are worth mentioning. The evaluation time of individual input fields is reported when Focus ▸ Output options ▸ Show timings is enabled. This includes the time that TEXMACS used in order to process the input and the output, so these timings should only be used as indications: benchmark tables should instead be computed directly inside the plug-in.

Computer algebra systems occasionally tend to generate large expressions that cannot be rendered on a single line. By default, such expressions are rewritten by TEXMACS so as to make line breaking possible. This means that wide fractions $\frac{a}{b}$ are allowed to be rewritten under the form a/b and similarly for square roots, matrices, etc. Although this is important for printing purposes, it also can make the output more difficult to read. If you are not interested in printing out your computations, then you may wish to turn off the rewriting mechanism, using Focus ▸ Preferences ▸ Do not break up large formulas.

11.1.4 Alternative session styles

When running sessions inside a laptop presentation, it is possible to opt for one of the fancier rendering styles. Figure 11.1 shows a basic alternate style that was selected using Focus ▸ Preferences ▸ Framed input fields. Similarly, Ring binder notebook style mimics the appearance of a notebook: see Figure 11.2.



Caas] $f(x) == \log\left(\log x - \frac{1}{x}\right) - \log\log(x+1);$

Caas] $f(\text{infinity}\ x)$

$$-2\frac{1}{\log(x)}\frac{1}{x}+\frac{1}{2}\frac{1}{\log(x)}\frac{1}{x^2}+\left(\frac{-1}{3}\frac{1}{\log(x)}-\frac{1}{2}\frac{1}{\log(x)^2}-\frac{2}{3}\frac{1}{\log(x)^3}\right)\frac{1}{x^3}+\left(\frac{1}{4}\frac{1}{\log(x)}+\frac{11}{24}\frac{1}{\log(x)^2}+\frac{1}{2}\frac{1}{\log(x)^3}\right)\frac{1}{x^4}+\left(\frac{-1}{5}\frac{1}{\log(x)}-\frac{5}{12}\frac{1}{\log(x)^2}+O\left(\frac{1}{\log(x)^3}\right)\right)\frac{1}{x^5}+O_x\left(\frac{1}{x^6}\right)$$

⇑ Linear algebra

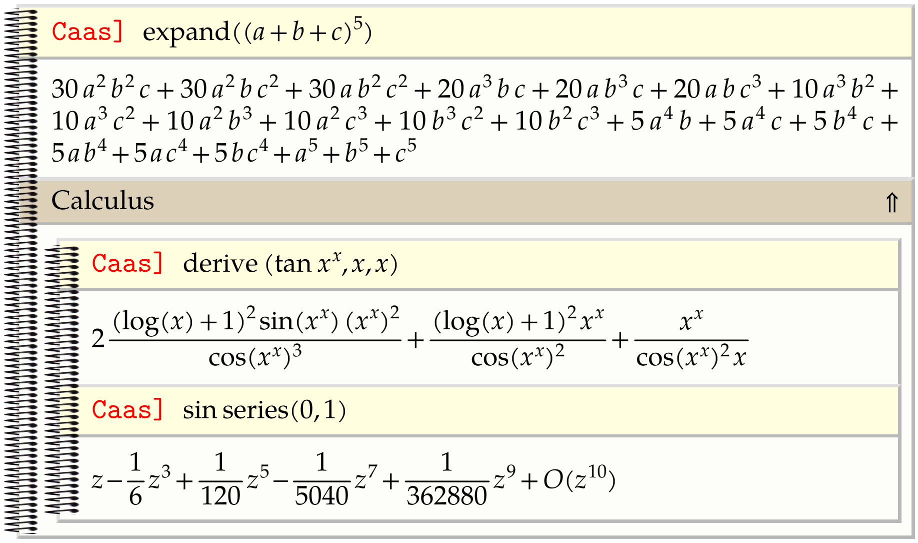
Caas] $\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$

$(c\,g+f\,h)\,(-a-e)+(a\,c+b\,f)\,g+(c\,d+e\,f)\,h+(a\,e-b\,d)\,i$

Caas] invert $\begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$

$$\begin{bmatrix} \frac{1}{a} & -\frac{b}{a\,c} \\ 0 & \frac{1}{c} \end{bmatrix}$$

Figure 11.1. The framed input style for computer algebra sessions.



Caas] expand$((a+b+c)^5)$

$30\,a^2\,b^2\,c+30\,a^2\,b\,c^2+30\,a\,b^2\,c^2+20\,a^3\,b\,c+20\,a\,b^3\,c+20\,a\,b\,c^3+10\,a^3\,b^2+10\,a^3\,c^2+10\,a^2\,b^3+10\,a^2\,c^3+10\,b^3\,c^2+10\,b^2\,c^3+5\,a^4\,b+5\,a^4\,c+5\,b^4\,c+5\,a\,b^4+5\,a\,c^4+5\,b\,c^4+a^5+b^5+c^5$

Calculus ⇑

Caas] derive $(\tan x^x, x, x)$

$$2\frac{(\log(x)+1)^2\sin(x^x)\,(x^x)^2}{\cos(x^x)^3}+\frac{(\log(x)+1)^2x^x}{\cos(x^x)^2}+\frac{x^x}{\cos(x^x)^2\,x}$$

Caas] sin series$(0,1)$

$$z-\frac{1}{6}z^3+\frac{1}{120}z^5-\frac{1}{5040}z^7+\frac{1}{362880}z^9+O(z^{10})$$

Figure 11.2. The ringbinder notebook style for computer algebra sessions.

These examples also demonstrate the use of subsessions. By pressing the ⇑ button (at the left-hand side in Figure 11.1 and at the right-hand side in Figure 11.2), you may fold the subsession; this also changes the symbol on the button to ⇓; a second click on this button unfolds the subsession.

11.1.5 Miscellaneous extras

TeXmacs provides facilities to write neat interfaces for computer algebra systems and other external scientific software. However, the extent to which these facilities are exploited by individual plug-ins rests on the shoulders of the developers of these plug-ins [18, 19, 20]. Multiline and mathematical input/output are examples of features that are exploited by several plug-ins, but not all. Let us describe a few others that were not mentioned so far.

Tab-completion

This mechanism allows for the automatic completion of long instructions using the tab key ⇥, after typing the first few characters. For instance, when typing `mathi` in a PARI/GP session and pressing ⇥, one obtains `mathilbert(|)`. The command `mathilbert` is used for the computation of Hilbert matrices:

```
Pari] mathilbert(3)
```

$$\%1 = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix}$$

Tab-completion is a nice tool not only for experienced users, but also for beginners who are still learning the precise names of potentially numerous commands. If there exist multiple completions, then pressing the tab key ⇥ several times allows you to cycle through the list of all possibilities.

Syntax highlighting

Modern program editors often use special colors for names of important instructions, comments, function declarations, and special constants. This technique of *syntax highlighting* makes programs more readable and several TeXmacs plug-ins support it. For instance, inside SCHEME sessions, commands that have actually been defined are rendered using a special color. This makes it possible to spot typos just by looking at the colors:

```
Scheme] (append (list 1 2) (list "a" "b"))

(1 2 "a" "b")
```

```
Eukleides
A B C right 5, 35 deg

draw
  (A.B.C)
  circle(A, B, C)
end

label
  A -135 deg
  B -45 deg
  C  45 deg
  A, B, C right
end
```

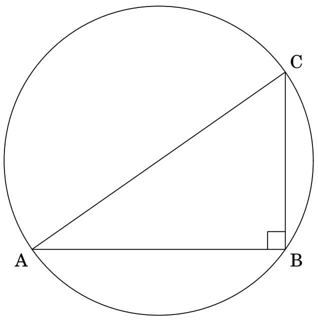


Figure 11.3. Example of an executable switch for the EUKLEIDES plug-in. On the left, we showed the unevaluated EUKLEIDES commands that describe the picture. On the right-hand side, you see the result after pressing `^*`. Hitting `^*` once more allows you to switch back to the source code.

```
Scheme] (appen (list 1 2) (list "a" "b"))  ;; Error!
Unbound variable: appen
```

11.2 Plug-ins as scripting languages

TEXMACS provides a few other types of interfaces to external systems besides the shell-like interfaces that we just described. First of all, it is possible to insert a so-called "executable switch" anywhere in the document using Insert ▸ Fold ▸ Executable.

For example, if MAXIMA is installed on your system, then Insert ▸ Fold ▸ Executable ▸ Maxima should yield something like Maxima . You may enter a MAXIMA expression in the yellow part of the switch, say Maxima `diff(x^x,x)`. Using `↩` or `^*`, you may now switch back and forth between the unevaluated input and the evaluated output $x^x(\log(x)+1)$. Using `⇧↩`, you enable multi-line input. This kind of executable switches are very useful for plug-ins like DRATEX, EUKLEIDES, FEYNMF, etc., which are mainly used for the efficient computation and insertion of special graphics inside TEXMACS documents: see Figure 11.3.

Some plug-ins such as MAXIMA can even be selected as a *scripting language* using Document ▸ Scripts ▸ Maxima. When doing so, MAXIMA can be used in the background as an assistant to the editor, by allowing you to operate directly on formulas. For instance, if your cursor is inside the formula $1+1$, then pressing `^↩` will automatically evaluate the formula using MAXIMA and replace it by the result 2.

When selecting a plug-in such as MAXIMA as the scripting language, a top-level menu Maxima appears. Common mathematical operations such as evaluation, simplification, differentiation, etc. thus become available through the menus. For example, the expression $(a+b+c)^5$ can be expanded by putting your cursor inside the formula and unleashing Maxima ▸ Simplification ▸ Expand. If you enable Maxima ▸ Keep evaluated expressions, then the evaluated expressions will be preserved. Expansion of the above expression then yields $\text{expand}((a+b+c)^5) = c^5+5\,b\,c^4+5\,a\,c^4+10\,b^2\,c^3+20\,a\,b\,c^3+10\,a^2\,c^3+10\,b^3\,c^2+30\,a\,b^2\,c^2+30\,a^2\,b\,c^2+10\,a^3\,c^2+5\,b^4\,c+20\,a\,b^3\,c+30\,a^2\,b^2\,c+20\,a^3\,b\,c+5\,a^4\,c+b^5+5\,a\,b^4+10\,a^2\,b^3+10\,a^3\,b^2+5\,a^4\,b+a^5$.

Basic executable switches as described above just allow for the evaluation of commands. If a plug-in can be used as a scripting language, then a more powerful variant of this mechanism becomes available, with the possibility to let the input of certain switches depend on the output of other ones. More precisely, assuming that you selected a scripting language from Document ▸ Scripts, you may insert a new *executable input field* using \ ! or Insert ▸ Link ▸ Executable input field. As before, when pressing ↩, the current input is evaluated and replaced by the corresponding output; you may switch back to the input by pressing ↩ once more.

Unlike executable switches, you may now attach an identifier to the executable input field by deactivating it or by editing the Ref field on the focus bar. Inside other executable input fields, you may then refer to the value of the field by inserting a *field reference* using \ ? or Insert ▸ Link ▸ Field reference. If you modify and re-evaluate an input field, TEXMACS will automatically re-evaluate all other fields that depend on it.

As an alternative to executable input fields, you may sometimes prefer to insert plain *input fields* using \ \ or Insert ▸ Link ▸ Input field. These fields can only be used for input and pressing ↩ inside such a field will only recompute those other fields that depend on it.

Example 11.3. Executable input fields can be used for educational purposes, by allowing parts of the document to be modified and recomputed by the reader. For instance, evaluation of

> The derivative of x^x equals diff(function,x).
>
> The second derivative is given by diff(derivative,x).

yields

> The derivative of x^x equals $x^x\,(\log(x)+1)$.
>
> The second derivative is given by $x^x\,(\log(x)+1)^2+x^{x-1}$.

The first occurrence of x^x is an input field with name `function`. Its first derivative is computed using an executable input field with name `derivative`. If the reader changes the input function x^x into something else and presses ↩, then both the first and second derivatives will be updated automatically.

11.3 Spreadsheets

Recent versions of TeXmacs include a rudimentary support for spreadsheets. The current implementation still requires speed optimizations, so it is only suitable for small tables that fit on a single page. One advantage with respect to more widely used systems such as Excel is the ability to perform the underlying mathematical computations using any plug-in that qualifies as a scripting language. Besides basic arithmetical operations, this makes it possible to perform more complex operations on cells such as symbolic differentiation or polynomial factorization.

From a technical point of view, spreadsheets are similar to the executable switches that were discussed in the previous section. The main difference is that the input fields are now organized in a table. Consequently, references indicate a row and a column rather than a name.

Assuming that you selected a scripting language in the menu Document ▸ Scripts, say Maxima, you may enter a new spreadsheet using Insert ▸ Table ▸ Textual spreadsheet or Numeric spreadsheet. You may edit the spreadsheet as an ordinary table, except that the ↵ key will attempt to reevaluate the cells of the table.

If the contents of a cell are preceded by =, then the cell will be considered as an input-output switch. More precisely, the input is a formula that will be evaluated using the current scripting language. After evaluation using ↵, only the result of the evaluation is shown in the cell. Pressing ↵ once more allows you to switch back and edit the input expression. Within formulas, one may refer to other cells using names such as c5 for the third row and the fifth column.

Example 11.4. On the left-hand side of the figure below, we have displayed a simple table with formulas for evaluating the sums of the first two items of each row. On the right-hand side, we have shown the result after evaluation.

1	10	=a1+b1
100	1000	=a2+b2

1	10	11
100	1000	1100

Figure 11.4. Evaluation of a simple spreadsheet.

Example 11.5. As illustrated in Figure 11.5, the cells may contain mathematical formulas and the spreadsheet may take advantage of any of the capacities of the scripting language.

$\sin(x^2)$
=diff(a1,x)
=diff(a2,x)
=diff(a3,x)

$\sin(x^2)$
$2x\cos(x^2)$
$2\cos(x^2)-4x^2\sin(x^2)$
$-12x\sin(x^2)-8x^3\cos(x^2)$

Figure 11.5. Computation of successive derivatives using Maxima.

TeXmacs supports a few special notations for applying operations on all cells in a subtable. Following EXCEL, you may use the notation c3:d5 for indicating all cells c3, c4, c5, d3, d4, d5 in the block from c3 to d5. An alternative notation ,..., for the range operator : can be entered by typing `,` `,`. In a similar way, you may enter the special notation $+\cdots+$ by typing `+` `+`. For instance, c3$+\cdots+$d5 stands for the sum of all cells between c3 and d5.

Example 11.6. The figure below shows how to sum the contents of a range of cells. Notice that each empty cell evaluates to zero.

15.10	15.10	30.20
100	125	75
28.50		14.25
12	16	20
=a1$+\cdots+$a4	=b1$+\cdots+$b4	=c1$+\cdots+$c4

15.10	15.10	30.20
100	125	75
28.50		14.25
12	16	20
155.6	156.1	139.45

Figure 11.6. Evaluation of sums of columns in a simple spreadsheet.

Note that copying and pasting of subtables works in the same way as for ordinary tables, with the additional property that names of cells are renumbered automatically, as well as any references to cells in the formulas. Automatic renumbering is also applied when inserting new columns or rows, or when removing existing ones.

We already mentioned the similarity between spreadsheets and executable switches. Both mechanisms can actually be combined in a natural way. On the one hand, references to executable switches (created using `\` `?`) can also be used inside spreadsheet cells. Conversely, each spreadsheet carries an invisible Ref field. If you select the spreadsheet, then you can edit this field on the focus bar. The Ref field of the spreadsheet is used as a prefix for referring to its cells from outside the table or from within other spreadsheets. For instance, if Ref equals sheet, then sheet-c4 refers to the field c4 inside the spreadsheet. The same mechanism applies if you need to refer to this cell in another spreadsheet.

11.4 Remote plug-ins

Plug-ins that have been installed on a remote computer can still be used inside TeXmacs over an SSH connection.

In order to make this work, you first have to make sure that SSH is installed on both computers and that connecting by SSH to the remote machine can be done automatically, without having to type a password. This can be done by copying your public identity on the local server to the remote server into the file `~/.ssh/authorized_keys`; see the documentation on SSH for more information.

The next step is to make sure that TEXMACS has been installed on both computers. The remote TEXMACS installation will mainly be used in order to detect those plug-ins that can be used on the remote computer.

When everything has been set up in this way, select Insert ▸ Session ▸ Remote in order to open the remote plug-in selector. Add the name of the remote server by typing its name or IP address and clicking on Add. After a small pause, the remote server should appear in the list together with the remote plug-ins that are supported. You may now simply select the plug-in you want to use from the list. Notice that remote plug-ins may take a few seconds in order to boot.

Servers that have been added to the list of remote plug-in servers will be remembered the next time you start TEXMACS. You may use the buttons Remove and Update in order to remove a server from the list and to update the list of supported remote plug-ins.

Chapter 12
Get it your way

12.1 Creating macros

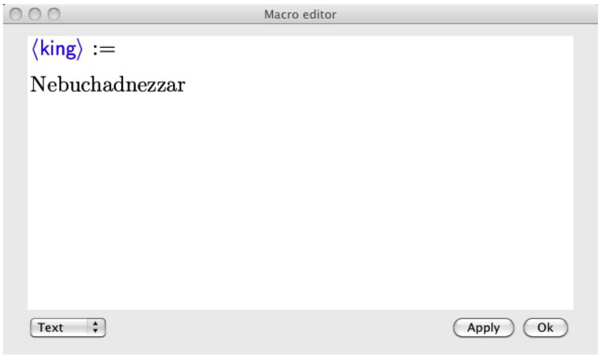
12.1.1 Hello Nebuchadnezzar

We have encountered an avalanche of markup elements throughout this book. Why add your own ones on top of the existing? One possible reason is that you may wish to save time by introducing abbreviations for lengthy names or notations. Let us see how to do this by defining your own *macros*.

Assume that we need the name of a king for our new novel and that Nebuchadnezzar seems to be the perfect candidate. Then it is natural to introduce a new macro king as an abbreviation for Nebuchadnezzar. The easiest way to do this is to open the "macro definition widget" using Tools ▸ Macros ▸ New macro. At the place of *enter-name*, you may enter the name of your macro: "king". The corresponding body "Nebuchadnezzar" can be typed below the := sign. When done (see Figure 12.1), simply click on Ok. Now that your macro has been defined, you can use it as many times as you wish by typing \ k i n g ↩:

> Hello Nebuchadnezzar!
> Nebuchadnezzar: my beloved leader.
> Bye, bye, Nebuchadnezzar.

One big advantage of using macros for abbreviations and special notations is that it suffices to modify the macro definitions whenever you change your mind: by positioning your cursor right after one of the king tags, you can edit the corresponding macro using Focus ▸ Preferences ▸ Edit macro. Just change the body of the macro to "Arikesari Maravarman Nindraseer Nedumaaran", if this is your new hero, and all Nebuchadnezzars will mutate accordingly.



⟨king⟩ :=
Nebuchadnezzar

Figure 12.1. The widget for macro definitions.

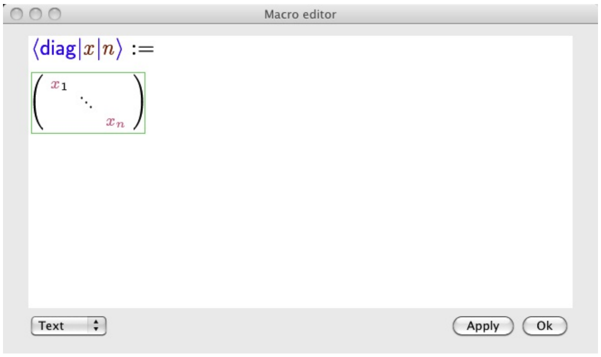

Macro editor
⟨diag|x|n⟩ :=
$\begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{pmatrix}$
Text
Apply
Ok

Figure 12.2. Example of a matrix definition with arguments.

12.1.2 Adding arguments

As our next example, assume that we wish to define a macro diag that produces a diagonal matrix

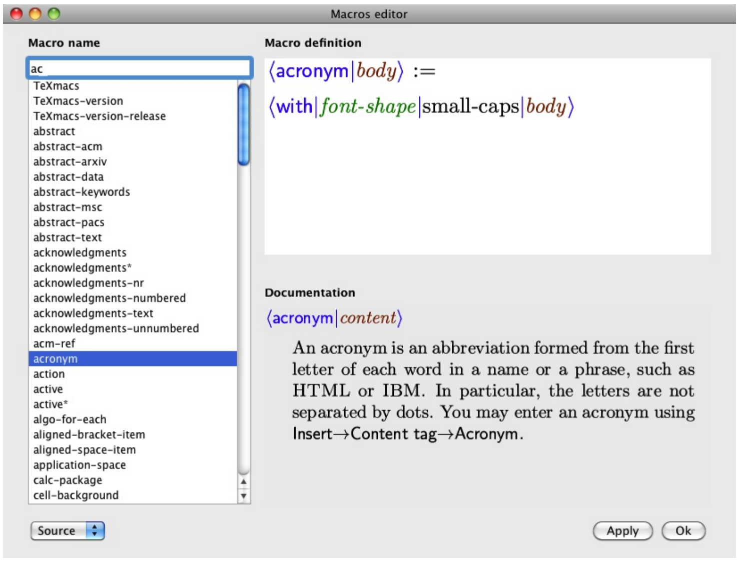
$$\begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{pmatrix}, \tag{12.1}$$

with the additional constraint that the symbols x and n can be changed each time the macro is used.

After entering the name of the macro as in the previous example, you may use the ⌥→ shortcut in order to insert additional macro arguments, say *x* and *n*. When entering the body of the macro, these arguments can be used inside the body by typing \ x ↩ and \ n ↩. The resulting macro definition should look like Figure 12.2. When applying the macro using \ d i a g ↩, an empty and editable slot is provided for each argument. Notice that only the first top-left occurrence of the parameter *x* can be edited, although the second one at the bottom-right will be updated accordingly during modifications.

The widget for macro editing is mainly intended for quick definitions of simple macros. In particular, you may miss the menus and the toolbars of the usual editor. Nevertheless, most of the usual keyboard shortcuts do work, as well as the contextual menus that appear when pressing the right mouse button (if your mouse has a single button, then you should hold the control key ^ while pressing that button). Whenever some keyboard shortcuts are preempted by the operating system, we also recall from section 2.3.5 that you may emulate the modifier keys ^, ⌥, and ⌘ using ⎋. For instance, ⌥→ is equivalent to ⎋ ⎋ →.

You may have noticed that we used the \ key both for applying the diag macro and for inserting the macro arguments *x* and *n*. The behavior of the \ key is context-dependent indeed. Inside the definition of a macro with arguments *x* and *n*, typing \ x ↩ will lead to the insertion of the macro argument *x*. In a context where the diag macro is defined, TeXmacs will automatically apply this macro when typing \ d i a g ↩. Notice that macro arguments are indicated



ac

TeXmacs
TeXmacs-version
TeXmacs-version-release
abstract
abstract-acm
abstract-arxiv
abstract-data
abstract-keywords
abstract-msc
abstract-pacs
abstract-text
acknowledgments
acknowledgments*
acknowledgments-nr
acknowledgments-numbered
acknowledgments-text
acknowledgments-unnumbered
acm-ref
acronym
action
active
active*
algo-for-each
aligned-bracket-item
aligned-space-item
application-space
calc-package
cell-background

⟨acronym|*body*⟩ :=

⟨with|*font-shape*|small-caps|*body*⟩

⟨acronym|*content*⟩

An acronym is an abbreviation formed from the first letter of each word in a name or a phrase, such as HTML or IBM. In particular, the letters are not separated by dots. You may enter an acronym using Insert→Content tag→Acronym.

Figure 12.3. Browsing the list of all macros using the macros editor.

using a brown color, whereas tags are displayed in blue.

Another technique that is useful for the definition of macros goes as follows. First design the prototype macro application in an ordinary TeXmacs window. That would simply be the diagonal matrix (12.1) for our example. Now copy this matrix and paste it inside the macro body when defining the diag macro. Finally replace the variables x and n by the macro arguments x and n that were entered using `\ x ↩` and `\ n ↩`. This copy-and-paste technique will turn out to be even more powerful when editing style files, as described later in this chapter.

12.1.3 Customizing an existing macro

Whenever your cursor is inside or just after a markup element, you may customize the corresponding macro definition using Focus ▸ Preferences ▸ Edit macro or 🔧 ▸ Edit macro. For example, assume that we entered the acronym GNU or LASER (using `\ a c r o n y m ↩` or Insert ▸ Content tag ▸ Acronym). If you wish to change the default rendering (a small capitals font), you can customize the corresponding acronym macro by positioning your cursor inside the text GNU or LASER and opening the macro editor with 🔧 ▸ Edit macro.

In certain cases, it may also be handy to browse the list of all existing macros that are active in your current style. This can be done by opening the macros editor via Tools ▸ Macros ▸ Edit macros: see Figure 12.3. You may find the acronym macro in the list at the left-hand side, possibly after typing the first few characters of the Macro name. A short description of the macro is displayed in the Documentation area, whenever available. The macro can be edited as usual in the Macro definition area.

In Figure 12.3, we selected "Source mode" for editing the body of the macro, as indicated by the pull-down menu at the bottom-left of the window. This mode is usually most convenient for editing complex macros, whereas the default text mode tends to be more suitable for simple, visually oriented macros. We also notice that the macro editor shows macro definitions in an abridged manner: inside style files and packages the definition from Figure 12.3 would read

```
⟨assign|acronym|⟨macro|body|⟨with|font-shape|small-caps|body⟩⟩⟩
```

Before we explain the presentation and syntax of source code in full detail, let us first analyze the customization of an existing macro on a more complex example.

12.1.4 Anatomy of a macro

The rendering of certain markup elements like acronym is quite straightforward. Many macros are far more complex and may recursively depend on other macros.

Assume for example that we wish to customize the proof environment using the macros editor and replace the "end-of-proof" symbol □ by ■. The internal structure of the proof is exposed by selecting Source instead of Text in the left bottom menu of the macros editor. For most styles, the source code is as follows:

```
⟨assign|proof|
    ⟨macro|body|
        ⟨render-proof|⟨proof-text⟩|body⟩⟩⟩
```

The prefix "render-" is used for submacros that specifically control the graphical layout. In this case, the arguments of render-proof are the (potentially translated) text "Proof" and the actual proof body. This additional level of indirection allows for the specification of alternative proof texts:

> **Continued proof of the master theorem.** We now conclude by applying the master lemma. □

When customizing a macro, it is important to first understand its structure and the intent of the other macros on which it relies. In many cases, you really want to customize one or more of these other macros. In our example, this leads us to examine the source code of the render-proof macro, which is given by

```
⟨assign|render-proof|
    ⟨macro|which|body|
        ⟨surround||⟨htab|0.5fn|0⟩⟨qed⟩|
            ⟨render-remark|which|
                body⟩⟩⟩⟩
```

The surround primitive provides a way to decorate large blocks of text (so-called "block markup") at the left and at the right. In this case, the proof text is decorated with the end-of-proof symbol □, which is produced using the macro qed. The symbol is preceded by a "horizontal tab" that flushes it to the right margin. It now suffices to change the definition of the qed macro to ■. The above proof then reads:

> **Continued proof of the master theorem.** We now conclude by applying the master lemma. ■

At this point, we succeeded in replacing the □ symbol by ■, but let us push our explorations a bit farther. The render-remark macro is another even more central rendering macro, which is also used for remarks, notes, examples, etc. Its source code is as follows:

```
⟨assign|render-remark|
    ⟨macro|which|body|
        ⟨render-enunciation|⟨remark-name|which⟨remark-sep⟩⟩|body⟩⟩⟩
```

As you can see here, render-remark depends on its turn on a yet more central macro render-enunciation, which is also used for theorems and exercises. This additional level of indirection is due to the fact that the bodies of major theorem-like enunciations are usually emphasized, whereas proofs, remarks, and other less prominent enunciations do not the change the font of the main body. Examination of the definition of render-remark also reveals the possibility to specify an alternative rendering for the names of remarks (e.g. REMARK 2.3) and the separator between this name and the corresponding body (usually a dot). Further investigations lead us to the source code of render-enunciation:

```
⟨assign|render-enunciation|
    ⟨macro|which|body|
        ⟨padded*|
            ⟨surround|which|⟨yes-indent*⟩|
                body⟩⟩⟩⟩
```

We have now reached the explicit graphical layout macro padded* that is used to add vertical whitespace before and after the enunciation.

```
⟨assign|padded*|
    ⟨macro|body|
        ⟨padded-normal|large-padding-above|large-padding-below|
            body⟩⟩⟩
```

The precise amount of spacing is controlled by the style parameters *large-padding-above* and *large-padding-below*. For all macros that depend on padded*, this leads to entries Focus ▸ Preferences ▸ Large padding above and Large padding below in the Style parameters group. If your cursor is inside a proof, then

$$\begin{tikzcd}
 & & & \text{proof} \arrow[dl] \arrow[d] & \\
 & & \text{proof-text} & \text{render-proof} \arrow[dl] \arrow[d] & \text{remark} \arrow[dl] \\
 & & \text{qed} & \text{render-remark} \arrow[dlll] \arrow[dl] \arrow[d] & \text{render-theorem} \arrow[dl] \\
\text{remark-name} & & \text{remark-sep} & \text{render-enunciation} \arrow[d] & \\
 & & & \text{padded*} \arrow[dlll] \arrow[dl] \arrow[d] & \\
\text{large-padding-above} & & \text{large-padding-below} & \text{padded-normal} &
\end{tikzcd}$$

Figure 12.4. The dependency tree for the proof macro.

you may change these style parameters. The padded-normal macro is defined directly in terms of typesetting primitives that cannot be further customized:

```
⟨assign|padded-normal|
    ⟨macro|before|after|body|
        ⟨surround|⟨vspace*|before⟩⟨no-indent⟩|⟨htab|0fn|first⟩⟨vspace|after⟩|
            body⟩⟩⟩
```

Figure 12.4 summarizes the recursive dependencies of the proof macro. Once again: before customizing an existing macro defined by the selected style, it is recommended to investigate its dependency tree and carefully identify those submacros that control the behavior that you wish to change.

12.2 Pilgrimage to the source

In the previous section, we have seen that it can be useful to select the "source code" rendering when editing macros. You may actually select this rendering style for any TeXmacs document using ⌘⌥S or Document ▸ Source ▸ Edit source tree, and for any selected portion of text using ⌘-. For certain documents, this gives a better grip on their full structure and in particular reveals any hidden information such as folded content.

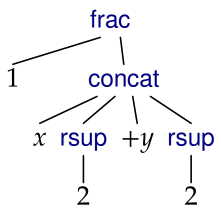
However, contrary to LaTeX and HTML (among others), the preferred representation of TeXmacs documents is *not* based on ASCII texts. This makes the concept of "source code" somewhat ill-defined for TeXmacs documents. As we will explain in this section, the best mental representation of a TeXmacs document is to regard it as an abstract tree. There are many ways to render such trees on a screen, some of which are closer to the printed end-result, and some of which expose more of the hidden internal structure.

12.2.1 Documents as trees

From a more conceptual perspective, TeXmacs systematically regards documents as *trees*. Consider for example the formula

$$\frac{1}{x^2+y^2}. \tag{12.2}$$

TeXmacs internally represents this formula by a tree[12.1]



(12.3)

The nodes of the tree are labeled by markup elements (internal primitives or user-defined macros) and the leaves of the tree are ordinary text strings.

Pursuing our conceptual perspective, one should distinguish between the abstract document tree and its graphical rendering. In particular, (12.3) is just an attempt to represent such a tree in a graphically pleasing way. The default rendering (12.2) is another representation and the "source code" rendering is yet another possible representation:

```
⟨frac|1|x⟨rsup|2⟩+y⟨rsup|2⟩⟩
```

The main virtue of "source code" with respect to other renderings is that it aims to make the full structure as transparent as possible. But even though TeXmacs proposes a default "source code" rendering, there is no intrinsic reason to prefer any particular such rendering over another one. For instance, scheme expressions provide an alternative that only involves plain ASCII text:

```
(frac "1" (concat "x" (rsup "2") "+y" (rsup "2")))
```

We will come back to this representation in chapter 14, since it is particularly useful when using SCHEME as an extension language for TeXmacs.

12.2.2 Selecting your preferred presentation of source code

The "source code" rendering of document trees can actually be customized via Document ▸ Source ▸ Preferences. We invite you to play around with the different possibilities in a document of your own (after enabling Document ▸ Source ▸ Source tree).

12.1. Notice that x^2 is represented as x⟨rsup|2⟩ and *not* as ⟨rsup|x|2⟩. This is more convenient for wysiwyg editing purposes, since you do not need to specify the precise mathematical expression that is being superscripted.

Angular

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            body⟩⟩⟩
```

Scheme

```
(assign "quick-theorem"
    (macro "body"
        (surround (no-indent)"Theorem. " ""
            (arg "body"))))
```

Functional

```
assign (quick-theorem,
    macro (body,
        surround (no-indentTheorem. , ,
            body)))
```

LaTeX

```
assign{quick-theorem}{
    macro{body}{
        surround{no-indentTheorem. }{}{
            body}}}
```

Figure 12.5. Different styles for rendering the same source tree.

First of all, you may choose your preferred "Main presentation style" among "angular", "scheme", "functional", and "LaTeX", as illustrated in the Figure 12.5.

Secondly, you may wish to reserve a special treatment for certain tags, such as the formatting tags concat and document for horizontal and vertical concatenation. In the menu Tags with a special rendering you may specify to which extent you want to treat such tags in a special way:

Raw. No tags receive a special treatment.

Format. All tags are rendered as source code, except for the formatting tags concat and document.

Normal. In addition to the formatting tags, a few other tags like compound, value, and arg are rendered in a succinct way.

Maximal. At the moment, this option is not yet implemented. The intention is to allow you to write your own customizations and to allow for a special rendering of basic mathematical operations such as plus.

These different options are illustrated in Figure 12.6.

Raw

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨document|
            ⟨surround|⟨concat|⟨no-indent⟩|The-
            orem. ⟩||
                ⟨arg|body⟩⟩⟩⟩⟩
```

Format

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            ⟨arg|body⟩⟩⟩⟩
```

Normal

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            body⟩⟩⟩
```

Maximal

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            body⟩⟩⟩
```

Figure 12.6. Different ways to render special tags.

None

```
⟨assign|
    quick-theorem|
    ⟨macro|
        body|
        ⟨surround|
            ⟨concat|
                ⟨no-indent⟩|
                Theorem. ⟩|
            |
            body⟩⟩⟩
```

Inline

```
⟨assign|
    quick-theorem|
    ⟨macro|
        body|
        ⟨surround|
            ⟨no-indent⟩Theorem. |
            |
            body⟩⟩⟩
```

Normal

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            body⟩⟩⟩
```

All

```
⟨assign|quick-theorem|⟨macro|body|⟨document|
⟨surround|⟨no-indent⟩Theorem. ||body⟩⟩⟩⟩
```

Figure 12.7. Different levels of compactification.

Another thing that can be controlled by the user is whether the presentation of tags should be compact or stretched out across several lines. In general, so-called *inline tags* are more readable when represented in a compact way. Inline tags are used for small pieces of content in the middle of a paragraph, such as a reference or an inline formula. On the other hand, *block tags* are used for longer pieces of content that may span over several paragraphs, such as proofs or item lists. In source code rendering, they typically look better when they are stretched out over several lines. Note that certain tags (like em for emphasized text) can be used both as inline and as block tags.

The Compactification menu proposes several levels of compactification:

None. The tags are all stretched out across several lines.

Inline. All non-inline tags are stretched out across several lines.

Normal. All inline arguments at the start of the tag are represented in a compact way. As soon as we encounter a block argument, the remainder of the arguments are stretched out across several lines.

Inline arguments. All inline arguments are represented in a compact way and only block tags are stretched out across several lines.

All. All source code is represented in a compact way.

The visual effect of the different levels of compactification is illustrated in Figure 12.7.

Finally, the Closing style menu allows you to specify the way closing tags should be rendered when the tag is stretched out across several lines. The rendering can either be minimalistic, compact, long, or recall the matching opening tag. The different options are illustrated in Figure 12.8.

Minimal

```
assign quick-theorem
    macro body
        surround ⟨no-indent⟩Theorem. |
            body
```

Compact

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            body⟩⟩⟩
```

Long

```
⟨assign|quick-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩Theorem. ||
            body
        ⟩
    ⟩
⟩
```

Repeat

```
⟨\assign|quick-theorem⟩
    ⟨\macro|body⟩
        ⟨\surround|⟨no-indent⟩Theorem. |⟩
            body
        ⟨/surround⟩
    ⟨/macro⟩
⟨/assign⟩
```

Figure 12.8. Different ways to render closing tags.

12.2.3 Data or program?

When designing your macros, you will quickly find out that TeXmacs macros have a dual character. Simple macros such as an abbreviation or a notation for diagonal matrices are usually visual in nature and best edited in the same way as ordinary text. More complex macros behave more like programs with local variables and subprograms. The source code representation tends to be more convenient for editing such macros. The borderline between textual data and macro programming is blurry: sometimes both aspects can be found in the same macro definition.

TeXmacs fully acknowledges this data-or-program duality and provides a few mechanisms to locally switch between normal and source style rendering. One typical use case is to quickly examine (and possibly modify) the "source code" of a small portion of text. For example, assume that we wish to study the markup that was used in order to produce "brown" and "jumps" in the fragment

> The quick brown fox $^{\text{jumps}}$ over the lazy dog.

Then it suffices to select "brown fox jumps" and press ⌘- (when activating the source tool using Tools ▸ Source macros tool, you may also use Source ▸ Activation ▸ Deactivate):

> The quick ⟨with|*color*|brown|brown⟩ fox ⟨move|jumps||0.5ex⟩
> over the lazy dog.

This technique inserts an additional (invisible) inactive* tag around the selection. You have to remove this tag in order to switch back to the original presentation, e.g. by placing your cursor after the "x" in "fox" and pressing ^⌫. Recall that the source code for the entire document can be edited using Document ▸ Source ▸ Edit source tree.

Conversely, when editing a macro definition in source mode, you may prefer the usual non-source rendering for the more graphical portions of text. Consider for instance the definition

⟨assign|*new-icon*|⟨macro|⟨icon|`tm_new_x4.png`⟩⟩⟩

You may prefer the following presentation that was obtained by selecting ⟨icon|`tm_new_x4.png`⟩ and pressing ⌘+:

⟨assign|*new-icon*|⟨macro|⟩⟩

A more complex example is the following:

⟨assign|*diag*|⟨macro|*var*|*dim*|$\begin{pmatrix} var_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & var_{dim} \end{pmatrix}$⟩⟩

Here we activated the usual rendering for the matrix using ⌘+, but switched back to source code rendering for the arguments *var* and *dim* using ⌘-.

12.2.4 The ASCII religion

Let us briefly discuss one aspect of source code that may not be of direct practical interest, but which may give you some insight into the design philosophy behind TeXmacs.

Programmers are accustomed to the use of ASCII as their privileged representation of source code. Over decades, many tools have been developed in order to make ASCII-style programming highly efficient. However, it is not so clear that a line of C++ code such as

```
P = 23*pow(x,3)*pow(y,2)*z + 17*x*pow(z,4)                (12.4)
```

is easier to read than

$$P=23x^3y^2z+17xz^4 \tag{12.5}$$

Similarly, some masochism is required to prefer LaTeX code

```
M = \left( \begin{array}{ccc}
  1 & \alpha_{1,2} & \alpha_{1,3}\\
  0 & 1 & \alpha_{2,3}\\
  0 & 0 & 1
\end{array} \right)                                        (12.6)
```

over

$$M=\begin{pmatrix} 1 & \alpha_{1,2} & \alpha_{1,3} \\ 0 & 1 & \alpha_{2,3} \\ 0 & 0 & 1 \end{pmatrix} \tag{12.7}$$

With an appropriate editor like TeXmacs, we do not only claim that (12.5) and (12.7) can be entered faster than (12.4) and (12.6): we may also regard them as a better way to represent "source code".

In fact, we regard it as an interesting challenge to develop a "source code editor" that is at least as efficient as traditional ASCII-based editors, but that allows for visually more attractive and readable presentations. The main advantage of existing editors is that the user is not constrained by the structure of a program. For instance, the following piece of code was made more readable through the manual insertion of spaces at appropriate places:

```
if (cond) hop   = 2;
else      holala= 3;
```

However, the precise formatting policy does not appear in the document, so manual intervention will be necessary if the variable `cond` is renamed `c`, or if the variable `holala` is renamed `hola`.

TeXmacs is not really intended to be a program editor yet, so no tools are provided for dealing with the above example in an automatic way. Nevertheless, the discussion applies in a lesser extent to the source code mode in TeXmacs. New users may feel somewhat constrained by the document structure and disoriented by the fact that ASCII-style editing habits not always apply. These disadvantages are still felt by more experienced users, but compensated by the benefits of the structured editing facilities. Nevertheless, it remains a challenge to make it even easier and more natural to edit source code.

12.3 Grouping your macros together

12.3.1 Preambles

The macro editor is an efficient tool for the definition of simple notations and for basic customizations of existing macros. The resulting macro (re-)definitions are automatically stored in the *preamble* of your document. More experienced TeXmacs users often do not use the macro editor and prefer to directly edit the preamble using ⌘⌥p or Document ▸ Part ▸ Show preamble. This has the advantage that you see all macro definitions at once. By putting them in a suitable order, you may also enhance their readability.

Preambles are edited much in the same way as ordinary documents, except that you are in source mode. Copying and pasting in particular continues to work as usual. This is a powerful technique for the definition of complex macros: first design a prototype macro application in an ordinary document, next copy and paste it into the body of your macro definition in the preamble, and finally perform the appropriate replacements of text by macro arguments.

Besides macros, you may also edit environment variables using the macro editor. Examples of environment variables are the page size, the left margin, the text color, or the number of the current subsection. Some of these environment variables can also be modified more directly using the Document menu. For example, the main text color for the document can be selected using Document ▸ Colors ▸ Foreground.

Just like the `\` key allows you to apply a macro whenever it is defined (see section 12.1.2), you may use it in order to obtain the value of any available environment variable. For example, `\` `c` `o` `l` `o` `r` `↩` yields the value of the current text color. Inside source code, TeXmacs uses a green color for such environment variables, as in *color*.

12.3.2 Style packages

The preamble contains style customizations that apply to one individual document. If you are particularly pleased with some of your customizations, then you may wish to bundle them in a *style package* that can be reused in other documents. One efficient way to start a new style package is Tools ▸ Macros ▸ Extract style package: this collects all existing customizations in your current document (both macros and environment variables) and puts them into a new style package.

Style packages are similar to usual TeXmacs documents, except that you edit them in source mode, just like preambles. When saving them to disk, you should also use the `.ts` extension instead of `.tm`. The native TeXmacs style packages are stored in the directory `$TEXMACS_PATH/packages`, where `$TEXMACS_PATH` points to the directory where you installed TeXmacs. You must put your own style packages in the same directory as the files that use them or in the directory `$TEXMACS_HOME_PATH/packages`, where `$TEXMACS_HOME_PATH` points to `~/.TeXmacs` on UNIX systems and to `C:\Documents and settings\`*`user_name`*`\Application Data\TeXmacs` on WINDOWS. After saving your style package in this way, you may apply it to any of your TeXmacs files using Document ▸ Style ▸ Add package ▸ Add other package.

Style files are a variant of style packages. Style files control the main document style, whereas style packages contain customizations that can be combined with other style files or packages. Style files must be complete in the sense that all major markup elements (such as section, theorem, equation, bibliography, strong, etc.) should be defined. Style packages typically contain extra personal notations or customizations of existing markup elements.

The complete style of your current document can be extracted using Tools ▸ Macros ▸ Extract style file. When saving style files to disk, you should again use the `.ts` extension. Native style files are stored in `$TEXMACS_PATH/styles`. You should put your own style files in `$TEXMACS_HOME_PATH/styles` or in the same directory as the files that use them. The main style is selected using Document ▸ Style and user-defined styles are automatically listed there.

12.4 The style-sheet language

So far, we have discussed how to introduce new notations and how to produce simple customizations of existing macros. For the design of more complex macros, $\text{T}_\text{E}\text{X}_\text{MACS}$ provides a special *style-sheet language*. This language consists of a set of special primitives that allow you to locally modify an environment variable, choose between several renderings, perform length computations, etc. In this section, we give an introduction to some of the most important primitives. A full exposition is beyond the scope of this book; for more details, we refer to the reference guide that is found in Help ▸ Reference guide.

Throughout this section, we assume the editor to be in "source mode", so that a top-level Source menu should be available.

12.4.1 Assignments

All user-defined $\text{T}_\text{E}\text{X}_\text{MACS}$ macros and style variables are stored in the "current typesetting environment". This environment associates tree values to string variables. Variables whose values are macros correspond to new primitives. The others are ordinary environment variables. The primitives for operating on the environment are available from Source ▸ Define.

You may permanently change the value of an environment variable using the assign primitive, as in the example

```
⟨assign|hi|⟨macro|Hi there!⟩⟩
```

You may also locally change the values of one or several environment variables using the with primitive:

```
⟨with|font-series|bold|color|red|Bold red text⟩
```

The with primitive can also be used for local redefinitions of macros:

```
⟨with | strong | ⟨macro | body | ⟨with | font-series | bold | color | red | body⟩⟩ | ⟨strong |
strong⟩ text⟩
```

The value of an environment variable may be retrieved using the value primitive, as in the following code to increase a counter:

```
⟨assign|my-counter|⟨plus|my-counter|1⟩⟩
```

Here we recall that the default rendering of ⟨value | *my-counter*⟩ inside source code is *my-counter* (see sections 12.2.2 and 12.3.1). Our example actually shows that both left-hand sides of assignments and values of environment variables are emphasized in green, which should help to locate environment variables inside source code. Similarly, macro arguments can be recognized by their brown color.

We recall that the value of an existing environment variable can be retrieved using the `\` key. For instance, `\ m y - c o u n t e r ↩` yields *my-counter*. In source mode, you may also use the keyboard shortcuts `⌘^=`, `⌘^w`, and `⌘^v` to produce assign, with, and value tags.

12.4.2 Macro expansion

The main interest of the style-sheet language is the possibility to define macros. These come in three flavors: ordinary macros, macros which take an arbitrary number of arguments and external macros, whose expansion is computed by SCHEME or a plug-in. The macro-related primitives are available from the Source ▸ Macro menu. Below, we will only describe ordinary macros.

Ordinary macros are defined using

```
⟨assign|my-macro|⟨macro|x1|···|xn|body⟩⟩
```

After such an assignment, my-macro becomes a new primitive with n arguments, which may be called using

```
⟨my-macro|y1|···|yn⟩
```

Inside the body of the macro, the arg primitive may be used to retrieve the values of the arguments to the macro. Notice that the default rendering of ⟨arg|*name*⟩ inside source code is *name*:

```
⟨assign|hello|⟨macro|name|Hello name, you look nice today!⟩⟩
```

It is possible to call a macro with fewer or more arguments than the expected number. Superfluous arguments are simply ignored. Missing arguments take the nullary uninit primitive (with rendering ?) as value:

```
⟨assign|hey|
    ⟨macro|first|second|
        ⟨if|
            ⟨equal|second|?⟩|
            Hey first, you look lonely today...|
            Hey first and second, you form a nice couple!⟩⟩⟩
```

You are allowed to use macros as values:

```
⟨assign|my-macro-copy|my-macro⟩
```

However, when using this style of macro programming, one should keep in mind that TEXMACS macros use a call-by-name evaluation strategy, contrary to functional programming languages such as SCHEME (see section 12.4.4 below). The compound tag may be used to apply macros that are the result of a computation:

```
⟨assign|overloaded-hi|
    ⟨macro|name|
        ⟨compound|
            ⟨if|⟨nice-weather⟩|happy-hi|sad-hi⟩|
            name⟩⟩⟩
```

12.4.3 Formatting primitives

Most $\TeX_{\text{MACS}}$ presentation tags can be divided in two main categories: *inline* tags and *block* tags. Inline tags are used for small pieces of text, whereas block tags can contain text that spans over several paragraphs. For instance, frac is a typical inline tag, whereas theorem is a typical block tag. Some tags (such as strong) are inline if their argument is inline and block otherwise.

The most primitive inline tag concat is used for horizontal concatenation: ⟨concat|⟨with|*color*|blue|blue⟩|⟨em|emphasis⟩⟩ is rendered as blue*emphasis*. Similarly, the most primitive block tag document is used for vertical successions of paragraphs: ⟨document|First|Second|Third⟩ is rendered as

First

Second

Third

The concat and document tags are so common that their names are actually hidden for the default rendering of source code (see section 12.2.2 and Figure 12.6).

When writing macros, it is important to be aware of the inline or block nature of tags, because block tags inside a horizontal concatenation are not rendered in an adequate way. For example, ⟨concat|Left|⟨document|Top|Bottom⟩|Right⟩ yields

```
LeftTop    Right
    Bottom
```

instead of the expected

LeftTop

BottomRight

If you need to surround a block tag with inline text, then you must use the surround primitive:

```
⟨assign|my-theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩⟨with|font-series|bold|Theorem. ⟩|⟨right-flush⟩|
            body⟩⟩⟩
```

In this example, we surrounded the body of the theorem with the bold text "**Theorem.**" on the left and a "right-flush" on the right-hand side. Flushing to the right is important in order to make the blue bounding box around the theorem look nice when editing the body of the theorem.

In most cases, TeXmacs does a good job in determining which tags are inline and which ones are not. However, you sometimes may wish to force a tag to be a block environment. For instance, the tag very-important defined by

```
⟨assign|very-important|⟨macro|body|⟨with|font-series|bold|color|red|body⟩⟩⟩
```

may be used both as an inline tag and as a block environment. When placing your cursor just before the with-tag and hitting ↩ followed by ⌫, you obtain

```
⟨assign|very-important|
    ⟨macro|body|
        ⟨with|font-series|bold|color|red|body⟩⟩⟩
```

These actions inserted a document tag around the body of the macro. The document tag itself is invisible (you should select Tags with special rendering ▸ raw in Document ▸ Source ▸ Preferences to make it visible), but its presence is indicated through the stretched rendering. Since the body of the macro is now a block, your tag very-important automatically becomes a block environment as well.

Another important property of tags is whether they contain normal textual content or tabular content. For example, consider the definition of the standard eqnarray* tag (with a bit of the presentation markup being suppressed):

```
⟨assign|eqnarray*|
    ⟨macro|body|
        ⟨with|par-mode|center|mode|math|math-display|true|par-sep|0.45fn|
            ⟨surround | ⟨no-page-break*⟩⟨vspace* | 0.5fn⟩ | ⟨vspace | 0.5fn⟩⟨no-
            indent*⟩|
                ⟨tformat|
                    ⟨twith|table-hyphen|y⟩|
                    ⟨twith|table-width|1par⟩|
                    ⟨twith|table-min-cols|3⟩|
                    ⟨twith|table-max-cols|3⟩|
                    ⟨cwith|1|-1|1|1|cell-hpart|1⟩|
                    ⟨cwith|1|-1|-1|-1|cell-hpart|1⟩|
                    body⟩⟩⟩⟩⟩
```

The use of surround indicates that eqnarray* is a block environment and the use of tformat specifies that it is also a tabular environment. Moreover, the twith and cwith are used to specify further formatting information: since we are a block environment, we enable hyphenation and let the table extend over the whole paragraph (unused space being equally distributed over the first and last columns). Furthermore, we have specified that the table contains exactly three columns.

Finally, it is important to bear in mind that style-sheets do not only specify the final presentation of documents on paper: they also determine the way documents are presented during the editing phase. Above, we have already mentioned the use of the right-flush tag in order to improve the rendering of visual border hints. Similarly, flags can be used to indicate the presence of invisible arguments while editing a document:

```
⟨assign|labeled-theorem|
    ⟨macro|id|body|
        ⟨surround|
            ⟨concat|
                ⟨no-indent⟩|
                ⟨flag|Id: id|blue|id⟩|
                ⟨with|font-series|bold|Theorem. ⟩⟩|
            ⟨right-flush⟩|
            body⟩⟩⟩
```

More generally, the specific tag with first argument “screen” may be used to display visual hints that are removed when printing the document (see section 7.6.2).

12.4.4 Evaluation control

The Source ▸ Evaluation menu contains several primitives to control the way expressions in the style-sheet language are evaluated. These primitives may in particular be used for the definition of “meta-macros” whose purpose is to define or redefine other macros. One typical example is the new-theorem meta-macro for the definition of new theorems:

```
⟨assign|new-theorem|
    ⟨macro|name|text|
        ⟨quasi|
            ⟨assign|⟨unquote|name⟩|
                ⟨macro|body|
                    ⟨surround|⟨no-indent⟩⟨strong|⟨unquote|text⟩. ⟩|⟨right-flush⟩|
                        body⟩⟩⟩⟩⟩⟩
```

When expanding ⟨new-theorem | theorem | Theorem⟩ in this example, we first evaluate all unquote instructions inside the quasi primitive, which yields the expression

```
⟨assign|theorem|
    ⟨macro|body|
        ⟨surround|⟨no-indent⟩⟨strong|Theorem. ⟩|⟨right-flush⟩|
            body⟩⟩⟩
```

Next, this expression is evaluated, thereby defining a macro theorem.

It should be noticed that the TEXMACS conventions for evaluation are slightly different then those from conventional functional languages such as SCHEME. We recall that SCHEME is the extension language for TEXMACS; see chapter 14 for a short introduction. The subtle differences between the style-sheet language and SCHEME are motivated by our objective to make it as easy as possible for the user to write macros for typesetting purposes. Assuming that you know the basics of SCHEME, it is instructive to examine the differences on a few examples.

When TEXMACS calls a macro ⟨macro|x_1|⋯|x_n|body⟩ with arguments `y1`,…,`yn`, the argument variables $x_1, \ldots, x_n$ are bound to the unevaluated expressions `y1`,…,`yn`, and the body is evaluated with these bindings. The evaluation of `yi` takes place each time the argument x_i is actually used during the evaluation of the macro. In particular, when applying the macro ⟨macro|*x*|*x* and again *x*⟩ to an expression `y`, the expression `y` is evaluated twice.

In SCHEME, the literal bodies of SCHEME macros are evaluated twice, whereas the arguments of functions are evaluated only once. On the other hand, when retrieving a variable (whether it is an argument or an environment variable), the value is not evaluated. Consequently, a TEXMACS macro

```
⟨assign|foo|⟨macro|x|⟨blah|x|x⟩⟩⟩
```

corresponds to a SCHEME macro

```
(define-macro (foo x)
  `(let ((x (lambda () ,x)))
     (blah (x) (x))))
```

Conversely, the SCHEME macro and function

```
(define-macro (foo x) (blah x x))
(define (fun x) (blah x x))
```

admit the following analogues in TEXMACS:

```
⟨assign|foo|⟨macro|x|⟨eval|⟨blah|⟨quote-arg|x⟩|⟨quote-arg|x⟩⟩⟩⟩⟩
⟨assign|fun|⟨macro|x|⟨with|x*|x|⟨blah|⟨quote-value|x*⟩|⟨quote-value|x*⟩⟩⟩⟩⟩
```

Here the primitives quote-arg and quote-value are used to retrieve the value of an argument or an environment variable. The TEXMACS primitives eval, quote, quasiquote, and unquote behave in the same way as their SCHEME analogues. The quasi primitive is a shortcut for quasi-quotation followed by evaluation.

12.4.5 Control flow

The Source ▸ Flow control menu contains TEXMACS analogues of basic control flow primitives that can be found in conventional programming languages: if, case, while, and for-each. However, be warned that the conditional constructs are quite fragile: they only apply to inline content and it is recommended that the various alternatives only affect the rendering, not the structure. For instance, consider the fragment

```
⟨assign|weird|
    ⟨macro|body|
        ⟨if|⟨my-condition⟩|Hi body!|⟨label|body⟩⟩⟩⟩
```

In the "then part" of the if primitive, the *body* argument is a regular macro argument that can be edited by the user (we say that it is *accessible*). In the "else part", the *body* argument is a hidden parameter that should be deactivated before you can edit it. To remain on the safe side, $\mathrm{T_EX_{MACS}}$ considers the *body* argument of the weird macro to be inaccessible, even when it is rendered according to the "then part".

The if primitive for conditional typesetting can be entered using `⌘^?` and is by far the most important control flow primitive. We recommend to mainly use it in combination with the compound primitive, as follows:

```
⟨assign|rich-appendix|
    ⟨macro|title|body|
        ⟨compound|
            ⟨if|⟨long-document⟩|chapter-appendix|section-appendix⟩|
            title|
            body⟩⟩⟩
```

In this example, rich-appendix is a block environment consisting of a title and a body, and that is rendered as a chapter for long documents and as a section for short ones. Notice that the following implementation would have been incorrect, since the if primitive currently only works for inline content:

```
⟨assign|rich-appendix|
    ⟨macro|title|body|
        ⟨if|
            ⟨long-document⟩|
            ⟨chapter-appendix|title|body⟩|
            ⟨section-appendix|title|body⟩⟩⟩⟩
```

The if primitive can also be used in order to implement optional arguments:

```
⟨assign|hey|
    ⟨macro|first|second|
        ⟨if|
            ⟨equal|second|?⟩|
            Hey first, you look lonely today...|
            Hey first and second, you form a nice couple!⟩⟩⟩
```

However, $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ is not clever enough to detect which arguments are optional and which arguments are accessible. This information should be specified manually using the drd-props primitive; see the reference guide for more information.

12.4.6 Computational markup

The menus Source ▸ Arithmetic, Text, Tuple, and Condition contain different primitives for computing with integers, strings, tuples, and boolean values. For instance, in the following code, the new-important tag defines a new "important tag" as well as a red variant:

```
⟨assign|new-important|
    ⟨macro|name|
        ⟨quasi|
            ⟨concat|
                ⟨assign|
                    ⟨unquote|name⟩|
                    ⟨macro|x|⟨with|font-series|bold|x⟩⟩⟩|
                ⟨assign|
                    ⟨unquote|⟨merge|name|-red⟩⟩|
                    ⟨macro|x|⟨with|font-series|bold|color|red|x⟩⟩⟩⟩⟩⟩⟩
```

Here we use the merge primitive in order to concatenate two strings. Precise positioning is possible through appropriate computations with lengths:

```
⟨assign|center-axis|⟨macro|body|⟨move|body||⟨minus|0.5ex|0.5h⟩⟩⟩⟩
```

The purpose of this macro is to vertically center the argument at the position of the fraction bar. This is achieved by moving it up by half the height `0.5ex` of an x-character and then back down by half the height `0.5h` of the argument itself (the length unit `h` is automatically defined to be the height of the main body inside the move primitive).

Chapter 13
Designing with style

The previous chapter gave a first introduction to the stylesheet language and how to add your own macros. We are now ready to dive deeper into this topic and investigate how to design complete document styles and style packages.

Scientific documents use a variety of style elements: section titles, item lists, theorems, headers, etc. New document styles are meant to offer the complete functionality of the built-in styles, but with a special design, which might for instance be dictated by the "graphical charter" of a journal. Since it would be cumbersome to redevelop all standard macros from scratch, new styles are typically designed as customizations of existing ones. The bulk of this chapter is dedicated to explaining how this works.

Style packages correspond to additional customizations that can be combined with any base style. This is typically useful for personal notations or when one or more macros need to be adapted in order to achieve a certain graphical effect. The explanations in this chapter may serve for the development of style packages that customize the standard styles.

Advanced users may also design style packages for new types of functionality such as literate programming or interactive courses. Such style packages are usually accompanied by extensions to the editor itself, in the form of keyboard shortcuts, special menu entries, or more complex editing routines. The next chapter gives a first glimpse of how to develop such enhancements.

13.1 Some words of caution

We recall that one of the main aims of beautiful typesetting and graphical design is to be "invisible": readers should focus on content and not be distracted by typesetting details. This is a quite delicate task which is really a job for experts on typography. Entire books have been written on this topic; see for instance [4, 40].

With the advent of typesetting software, the bulk of the traditional design tasks reduces to the development of high quality document styles. *A priori*, this still requires the same kind of expertise, so why would you want to this yourself? One reason could be that you wish to reproduce the style of a particular journal. But maybe you just got bored aping your colleague next door and prefer to have some fun creating your own distinctive style instead.

If you go down this road and design your own style, then it is tempting to introduce all kinds of fancy decorations for standard environments, such as boxes around theorems, drop shadows for section titles, your favorite pictures for list items, etc. Keep in mind that such phantasies may quickly saturate and irritate your readers. Instead, we invite you to search for more subtle ways to distinguish yourself. This can often be achieved through the careful selection of a few fonts, or by modifying some spacings or separators. For example, the simple use of a sans serif font for theorems will already distinguish your style from all built-in TeXmacs styles:

Theorem 13.1. *This statement has a distinctive look & feel.*

Notice that it is not necessary to refrain from all forms of ornaments. For instance, it may be all right to use the following type of main section titles:

——— My fancy section ———

Although such titles are somewhat "aggressive", they tend to occur only a few times in a document. As long as this is the only fantasy in your document style, this remains acceptable, while immediately ensuring a distinctive look and feel.

Another thing that you should worry about is related to the dual "data versus program" nature of TeXmacs macros that we first mentioned in section 12.2.3. Now the programming side becomes more significant when macros are grouped together in style files or packages. In many respects, the development of style files is analogous to writing software libraries, with macros in the role of library functions and subroutines. Besides aesthetical considerations, it is therefore important to think about issues like dependencies, maintainability, and orthogonality (i.e. how well do your macros combine with macros from other packages?).

Once again, you should be guided by the notion of simplicity. The simpler your personal customizations are and the more they rely on standard mechanisms, the more likely they will continue to work well with future versions of TeXmacs and in combination with other style packages. For example, if your macros crucially depend on some obscure internal macro that you found in one of the TeXmacs style packages, then they will break down if the internal macro happens to disappear in a future version of TeXmacs. Similarly, your macros may fail to work in combination with alternate styles that do not implement the required internal macro. For similar reasons, you should avoid complex macros that adjust the typesetting process in subtle ways: ongoing progress in the TeXmacs typesetter may make it difficult to maintain such macros.

13.2 Anatomy of a style package

Before you start designing your own style files, we recommend you take a closer look at the existing style files. One fairly complete example is the `svjour` base style for articles published by SPRINGER VERLAG. This style is selected using Document ▸ Style ▸ Article ▸ Springer ▸ svjour. After selecting this style, you may open the corresponding source file using Document ▸ Style ▸ Edit style.

The first line of the `svjour` style file specifies the style packages on which the style is based. It essentially uses the same style packages as the standard `article` style, together with one additional package `std-latex`:

```
⟨use-package|std|env-base|env-math|env-float|env-program|header-article|
title-base|section-article|std-latex⟩
```

The `std-latex` package is useful when adapting a LATEX style file to TEXMACS, since it provides several macros that allow you to specify global style parameters in "the LATEX way". For instance, the `svjour` style uses the following lines to specify the global margins and text width—both for single and double column articles:

```
⟨assign|tex-odd-side-margin|⟨macro|⟨if|⟨equal|par-columns|1⟩|0pt|-30pt⟩⟩⟩
⟨assign|tex-even-side-margin|⟨macro|⟨if|⟨equal|par-columns|1⟩|0pt|-30pt⟩⟩⟩
⟨assign|tex-text-width|⟨macro|⟨if|⟨equal|par-columns|1⟩|25.5cc|17.8cm⟩⟩⟩
```

Various other global style parameters are specified in a similar way.

The remainder of the `svjour` style file redefines some of the most significant macros that determine the layout of articles. One may distinguish the following main categories:

- Specification of the fonts and font sizes to be used.
- Rendering of sectional macros.
- Specification of headers and footers to be used.
- Theorem-like environments.
- Item lists and enumerations.
- Rendering of the document title and the abstract.

Let us briefly study how some of these customizations were carried out for `svjour`. More general and detailed explanations will be provided in the sections below.

Fonts and font sizes

Style files for journals often specify the main document font and its variants with care. The `svjour` style specifies the default font size to be `10pt` with a `0.2em` interline space, but also redefines the standard tags for other font sizes, such as small, large, smaller, etc. For example, the default rendering of footnotes uses a smaller font size that is specified through the smaller tag. The `svjour` style redefines it to use a `9pt` font with a `2pt` interline space:

```
⟨assign|smaller|⟨macro|x|⟨with|font-base-size|9|par-sep|2pt|x⟩⟩⟩
```

Sectional macros

After the main document font and page layout, the rendering of section titles is the next most significant characteristic of a scientific document style. Most journals use bold, emphasized, or small capital fonts, and carefully specify the amount of vertical space to be inserted before and after titles. Styles that were adapted from LaTeX by means of the `std-latex` package may reproduce TeX-style rubber spaces using the tex-len macro. For instance, `svjour` renders main section titles in the following way:

```
⟨assign|section-title|
    ⟨macro|name|
        ⟨sectional-normal-bold|
            ⟨concat|
                ⟨vspace*|⟨tex-len|21dd|4pt|4pt⟩⟩|
                ⟨normal-size|name⟩|
                ⟨vspace|⟨tex-len|10.5dd|4pt|4pt⟩⟩⟩⟩⟩⟩
```

Here sectional-normal-bold is a utility macro for rendering bold section titles. It also takes care of a few other issues such as forbidding page breaks just after the title.

Headers and footers

Most styles determine headers and footers as a function of the title and names of the authors. TeXmacs provides a few "call-back macros" for this purpose: header-title (document title), header-author (document author), header-primary (primary sections), and header-secondary (secondary sections). For example, the header-author macro is called with the name of the author as an argument, when specifying this information in the title of your document. The `svjour` style exploits this mechanism to redefine the even page header so as to contain the author's name:

```
⟨assign|header-author|
    ⟨macro|name|
        ⟨assign|page-even-header|
            ⟨small|
                ⟨no-indent⟩⟨page-number⟩⟨htab|5mm⟩name⟩⟩⟩⟩
```

Theorem-like environments

Scientific articles require many technical markup elements like theorems, enumerations, algorithms, technical pictures, etc. Full-fledged styles may redefine the rendering of such markup elements. The standard TEXMACS styles contain a large number of macros that allow you to customize specific rendering aspects. For instance, the main rendering of theorem-like environments is controlled through the render-theorem macro. But it is also possible to change the font for the text "THEOREM 3.6." using theorem-name, as well as the separator "." after the number using theorem-sep. For our example `svjour` style, we essentially have

```
⟨assign|theorem-name|⟨macro|name|⟨with|font-series|bold|name⟩⟩⟩
⟨assign|theorem-sep|⟨macro|. ⟩⟩
⟨assign|render-theorem|
    ⟨macro|which|body|
        ⟨padded-normal|1fn|1fn|
            ⟨surround|⟨theorem-name|which⟨theorem-sep⟩⟩||
                ⟨with|font-shape|italic|body⟩⟩⟩⟩⟩
```

In the definition of render-theorem, the padded-normal macro is used for specifying the vertical padding around theorem-like environments. The `std-utils` style package contains various other handy utility macros of this kind.

Lists and enumerations

Different nesting levels of list environments typically require separate rendering styles. TEXMACS provides the macros enumerate-1, enumerate-2, etc. for top-level enumerations, second level sub-enumerations, and so on. Most style files limit themselves to three nesting levels; for `svjour`, we have

```
⟨assign|aligned-accolade-item|⟨macro|x|⟨aligned-item|⟨with|font-shape|right|
{x}⟩⟩⟩⟩
⟨new-list|enumerate-1|aligned-dot-item|identity⟩
⟨new-list|enumerate-2|aligned-accolade-item|⟨macro|x|⟨number|x|alpha⟩⟩⟩
⟨new-list|enumerate-3|aligned-dot-item|⟨macro|x|⟨number|x|roman⟩⟩⟩
```

The new-list construct is used to define the macros enumerate-1, enumerate-2, and enumerate-3; its second argument is a macro that is used for rendering items; the third argument contains a macro that is used for transforming the numbers of items. The macros aligned-item and aligned-dot-item are standard rendering macros that can be redefined. The rendering of lists themselves is controlled through the render-list macro.

Document titles and abstracts

One of the most complex parts of style files concerns document titles and abstracts. The difficulty stems from the fact that title and abstract information should really be thought of as a collection of metadata for the document. The rendering then proceeds in two steps: one first has to collect and reorganize the data in an appropriate way and then call lower-level rendering macros in an appropriate order. Here one has to keep in mind that a lot of information (keywords, subject classifiers, affiliations, etc.) is optional or may need to be recombined (various authors with multiple affiliations).

It is recommended that style files only customize the final rendering macros for title and abstract information. The title mainly consists of a succession of "blocks" that are rendered using the doc-title-block macro. Specific kinds of information are rendered using dedicated macros doc-render-title, doc-subtitle, doc-date, etc. For example, the rendering of the main document title is defined as follows for the `svjour` style:

```
⟨assign|doc-render-title|
    ⟨macro|x|
        ⟨surround||⟨vspace|11.24pt⟩|
            ⟨doc-title-block|⟨larger|⟨with|math-font-series|bold|font-series|bold|
            x⟩⟩⟩⟩⟩⟩
```

13.3 Some tips before things get technical

Before going deeper into the technical details about how to control specific style elements, let us first discuss a few general tips for the development of style packages.

Which macros to redefine

Most standard style files and packages consist of a succession of macro definitions and specifications of default values for environment variables. When customizing an existing style, it is important to carefully select the macros and environment variables to be redefined.

Now certain tags in the standard style files are directly exposed to the end-user through the interface. For instance, you have direct access to the tags strong and section through the menu entries Insert ▸ Content tag ▸ Strong and Insert ▸ Section ▸ Section. Other tags such as section-title or render-theorem are rather provided for customization purposes by developers of style files. Some style packages also define auxiliary helper macros for internal purposes.

Although sufficiently simple tags like strong can be redefined directly in your own style files, this is not recommended for more complex tags such as section. Indeed, the section tag takes care of many tasks: rendering the title, resetting the counter for subsections, entering the title into the table of contents, and so on. In order to customize complex tags of this kind, you should rather redefine one or more of the "companion tags" section-title, section-clean, section-toc, etc. that control these more specific subtasks.

Customizing existing macros

Imagine that you want to change the rendering of a given tag, like lemma. We have just seen that $\TeX_{\text{MACS}}$ provides a set of carefully designed companion macros that can be customized to modify specific aspects of the rendering. For instance (see section 13.2), you are supposed to redefine one of the macros render-theorem, theorem-name, or theorem-sep in order to customize the rendering of lemma and all other theorem-like environments.

However, in some cases, it may not be clear which companion macro to customize. If we just wanted to change the presentation of lemmas and not of any other theorem-like environments, then we clearly cannot modify render-theorem, theorem-name, or theorem-sep. Besides, you may not want to spend too much time on understanding the macro hierarchy of $\TeX_{\text{MACS}}$, thereby ignoring the very existence of render-theorem, theorem-name, and theorem-sep.

So imagine that you want all lemmas to appear in red. One thing you can always do is copy the original definition of the lemma macro to a safe place and redefine it on top of the original definition:

```
⟨assign|orig-lemma|lemma⟩
⟨assign|lemma|⟨macro|body|⟨with|color|red|⟨orig-lemma|body⟩⟩⟩⟩
```

Alternatively, if only the text inside the lemma should be rendered in red, then you may do:

```
⟨assign|orig-lemma|lemma⟩
⟨assign|lemma|⟨macro|body|⟨orig-lemma|⟨with|color|red|body⟩⟩⟩⟩
```

However, be warned that this mechanism is somewhat fragile: if the name orig-lemma was already in use (for instance, if you already performed a similar customization in another style package), then you may introduce a circular dependency lemma → orig-lemma → lemma → orig-lemma → ⋯. Here we note that $\TeX_{\text{MACS}}$ contains no safeguards against programming errors by developers of style packages: infinite loops of the above type will simply crash the editor.

One obvious fix is to choose the backup name orig-lemma with more care; e.g. uncolored-lemma might be more appropriate. It is also a good idea to define the backup macro if only if no macro with the same name already exists; this can be done by using the provide primitive instead of assign:

```
⟨provide|uncolored-lemma|lemma⟩
⟨assign|lemma|⟨macro|body|⟨with|color|red|⟨uncolored-lemma|body⟩⟩⟩⟩
```

At the end of section 13.7, the redefinition of inc-theorem shows a more robust but subtle mechanism for customizing existing macros.

Local customizations

Another frequent situation is that you only want to modify the rendering of a tag when it is used inside another one. On the web, the *Cascading Style Sheet* language (CSS) provides a mechanism for doing this. In TeXmacs, you may simulate this behavior by redefining macros inside a with. For example, imagine that we want to suppress the inter-paragraph space inside lists within theorem-like environments. Then we may use:

```
⟨assign|orig-render-theorem|render-theorem⟩
⟨assign|render-theorem|
    ⟨macro|name|body|
        ⟨with|orig-render-list|render-list|
            ⟨with|render-list|⟨macro|x|⟨orig-render-list|⟨with|par-par-sep|0fn|x⟩⟩⟩|
                ⟨orig-render-theorem|name|body⟩⟩⟩⟩⟩
```

On the one hand, this mechanism is a bit more complex than CSS, where it suffices to respecify the *par-par-sep* attribute of lists inside theorems. On the other hand, it is also more powerful, since the render-theorem macro applies to all theorem-like environments at once.

Rely on the "standard utilities"

The style package `std-utils` contains various useful "helper macros" that should make it easier to develop style packages. It mainly contains macros for

- Writing block environments that extend over the entire paragraph width. Notice that the `title-base` package provides some additional macros for wide section titles.
- Writing wide block environments that are padded, underlined, overlined, or in framed boxes.
- Recursive indentation.
- Setting page headers and footers.
- Localization of text.

It is good practice to rely on these standard macros whenever possible. Indeed, the low-level TeXmacs internals may be subject to minor changes. When building upon standard macros with a clear intention, you increase the upward compatibility of your style-sheets.

Internationalization

Certain tags like theorems, figures, or bibliographies need to print English text like "Theorem", "Figure", or "References". In order to allow this text to be customized, such tags usually come with companion macros theorem-text, figure-text, bibliography-text, etc. For example, the `svjour` style redefines the figure-text macro to print "Fig." instead of "Figure". In order to automatically translate the text from English into the current language, you should pass it as an argument to the localize macro. For instance, the `section-base` package contains the following default definition of bibliography-text:

```
⟨assign|bibliography-text|⟨macro|⟨localize|Bibliography⟩⟩⟩
```

The directory `$TEXMACS_PATH/langs/natural/dic` contains the dictionaries for translations into the supported languages. Here `$TEXMACS_PATH` is the place where TeXmacs is installed on your system.

Converting LaTeX styles

Style files for TeX/LaTeX are particularly complex and ill-behaved, with lots of auxiliary macros and style parameters without clear semantics. For this reason, the built-in LaTeX converters perform rather poorly on style files. TeXmacs provides equivalents for some of the most important LaTeX styles. If you wish to mimic another style, then we recommend that you proceed as follows:

- Try to import the LaTeX style file. Most of the structure will be lost, but the converter sometimes manages to import at least a few macros and environment variables. The result of the conversion may therefore provide a reasonable start for the development of a TeXmacs equivalent for your LaTeX style.

- Determine an existing TeXmacs style that comes as close as possible to the desired style and base your new style on it using use-package. Notice that the converter from the previous step may already have come up with a good proposal.

- Add a line ⟨use-package|std-latex⟩ to your style file and customize the utility macros from that package in order to specify the most important layout parameters.

- Patiently customize various other macros in order to match the desired style, following the mechanisms that will be described in the subsections below.

Concerning the third step, we notice that TeXmacs does not always use the same conventions as TeX/LaTeX when it comes to global layout parameters. For example, the style parameters `\oddsidemargin` and `\evensidemargin` for left margins on odd and even pages do not mean what you would think, since TeX automatically adds one extra inch. The `std-latex` package provides macros tex-odd-side-margin and tex-even-side-margin that mimic this somewhat twisted behavior in TeXmacs. Various other style parameters `\textwidth`, `\topmargin`, `\jot`, `\abovedisplayskip`, etc. admit similar analogues tex-text-width, tex-top-margin, tex-jot, tex-above-display-skip.

The `std-latex` package also defines a macro tex-len with three arguments `default-length`, `plus`, and `minus` that emulates the syntax of TEX rubber lengths. For example, ⟨tex-len|1 em|0.5 em|0.25 em⟩ stands for the rubber length with default, minimal, and maximal values `1 em`, `0.75 em`, and `1.5 em`, respectively. The corresponding TEXMACS syntax for this rubber length is ⟨tmlen | 0.75 em|1 em|1.5 em⟩. See pages 62 and 7.1 for more information on TEXMACS lengths.

13.4 Customizing the general page layout

The general layout of a document is mainly modified by setting the appropriate environment variables for page layout and paragraph layout (see the reference manual for a complete list of these variables). For example, by including the following lines in your style file, you can set the page size to `letter` and the left and right margins to `1.25 in`:

```
⟨assign|page-type|letter⟩
⟨assign|page-odd|1.25in⟩
⟨assign|page-even|1.25in⟩
⟨assign|page-right|1.25in⟩
```

Recall from section 3.10 that the margins may be different on even and odd pages; the environment variables *page-odd* and *page-right* correspond to the left and right margins on odd pages.

Note that the environment variables for page layout are quite different in TEXMACS and TEX/LATEX. In order to make it easier to adapt LATEX style files to TEXMACS, we have therefore provided the `std-latex` package, which emulates the environment variables of TEX/LATEX. Typically, this allows you specify the global layout using declarations such as

```
⟨assign|tex-odd-side-margin|⟨macro|20pt⟩⟩
⟨assign|tex-even-side-margin|⟨macro|20pt⟩⟩
⟨assign|tex-text-width|⟨macro|33pc⟩⟩
```

Notice that *page-odd*, *page-even*, etc. are lengths, whereas tex-odd-side-margin, tex-even-side-margin, etc. are macros that return a length.

The page headers and footers are usually not determined by global environment variables or macros, since they may change when a new chapter or section is started. Instead, TEXMACS provides the call-back macros header-title, header-author, header-primary, and header-secondary. These macros get called when the document title or author are specified, or when a new primary or secondary section is started (by default, primary sections correspond to chapters in books, and to sections in articles). For instance, the following redefinition makes the principal section name appear on even pages, together with the current page number and a wide underline.

```
⟨assign|header-primary|
    ⟨macro|title|nr|type|
        ⟨assign|page-even-header|
            ⟨quasiquote|
                ⟨wide-std-underlined|
                    ⟨page-the-page⟩⟨htab|5mm⟩⟨unquote|title⟩⟩⟩⟩⟩⟩
```

13.5 Processing title information

The "titles" of scientific documents usually carry a lot of additional information about the authors, their affiliations, the creation date, grant acknowledgments, etc. For this reason, $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ treats the title information as a miniature database, and the graphical rendering proceeds in two phases: the most relevant information is first extracted from the database and reorganized. The actual rendering is done at a second stage, using dedicated macros for this purpose.

The first stage is fairly complex, since one has to deal with various optional data fields and potentially multiple authors with multiple affiliations. Various styles may present the data in different orders and one has to decide how to present common affiliations among coauthors (see section 3.3.1 for a few common styles). Since the required rewritings are rather intricate, they are not performed by $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ macros, but rather through "external" SCHEME routines whose precise description falls beyond the scope of this book.

The second stage is more straightforward: for each kind of title information, there is a corresponding rendering macro that can be customized by particular styles. For instance, the main title, an optional subtitle, the creation date, and miscellaneous extra title fields are rendered using the macros doc-title, doc-subtitle, doc-date, and doc-misc. So if the date should appear in a bold italic typeface and at a distance of at least `0.5fn` from the other title fields, then you may redefine doc-date as

```
⟨assign|doc-date|
    ⟨macro|x|
        ⟨surround|⟨vspace*|0.5fn⟩|⟨vspace|0.5fn⟩|
            ⟨doc-title-block|⟨with|font-shape|italic|font-series|bold|x⟩⟩⟩⟩⟩
```

The helper macro doc-title-block should be used for rendering atomic blocks of title information; many styles implement this macro by centering title blocks, whereas other styles rather align them to the left. $\text{T}_{\text{E}}\text{X}_{\text{MACS}}$ also uses the macro doc-make-title for encapsulating all title information. You may specify an amount of padding between titles and the main text by customizing this macro.

The rendering of author information is done using similar macros author-name, author-affiliation, author-email, author-homepage, and author-misc. These macros should rely on the macro doc-author-block for rendering atomic blocks of author information. In addition, TeXmacs provides variants author-affiliation-note, author-email-note, author-homepage-note, and author-misc-note that are used when several authors share common information. These variants take one additional argument that contains the symbol (such as †) that is used to link the shared information to the corresponding authors.

Author information is often rendered differently for documents with one versus several authors. In case of a single author, the doc-author macro is used for rendering the block with all author information. This macro behaves similarly to doc-title, doc-subtitle, etc., and should rely on doc-title-block for its rendering. If there are multiple authors, then the doc-authors macro is used for rendering the complete list of authors (the macro uses one argument for each author), whereas the information of each individual author is encapsulated inside a block that is rendered using the doc-authors-block macro.

Some further global metadata are provided in the abstract rather than in the title. The rendering of the abstract is controlled via the macro render-abstract. In addition, TeXmacs provides the macro abstract-keywords for rendering a list of keywords (one argument for every keyword) and the macros abstract-acm, abstract-msc, and abstract-pacs for specifying ACM, AMS, and PACS subject classifiers.

13.6 Customizing sectional tags

TeXmacs provides the same sectional tags as LaTeX: part, chapter, section, subsection, subsubsection, paragraph, subparagraph, and appendix. TeXmacs also implements the unnumbered variants part*, chapter*, etc. and special section-like tags bibliography, table-of-contents, the-index, the-glossary, list-of-figures, list-of-tables.

One important additional “predicate macro” is sectional-short-style. If it evaluates to `true`, then appendices, tables of contents, etc. are considered to be at the same level as sections. In the contrary case, they are at the same level as chapters. Typically, articles use the short sectional style whereas books use the long style.

The rendering of a sectional tag *x* is controlled through the macros *x*-sep, *x*-post-sep, *x*-title and *x*-numbered-title. The *x*-sep macro prints the separator between the section number and the section title. It defaults to the macro sectional-sep, which defaults in its turn to a wide space. For example, after redefining

```
⟨assign|sectional-sep|⟨macro| – ⟩⟩
```

sectional titles typically look as follows:

13.1 - Hairy GNUs

Similarly, *x*-post-sep prints the separator between the section title and subsequent text. The *x*-title and *x*-numbered-title macros respectively specify how to render unnumbered and numbered section titles. Usually, the user only needs to modify *x*-title, since *x*-numbered-title is based on *x*-title. However, if the numbers have to be rendered in a particular way, then it may be necessary to redefine *x*-numbered-title. For instance, consider the redefinition

```
⟨assign|subsection-numbered-title|
    ⟨macro|name|
        ⟨sectional-normal|
            ⟨with|font-series|bold|⟨the-subsection⟩. ⟩name⟩⟩⟩
```

This has the following effect on the rendering of subsection titles:

2.3. Very hairy GNUs

Notice that the sectional-normal macro comes from the `section-base` style package. You may find several similar macros sectional-normal-italic, sectional-centered-bold, etc. there for some of the most frequent ways to render section titles.

There are two main rendering styles for sectional titles. By default, paragraphs and subparagraphs use a "short" rendering style, with a body that starts immediately after the title:

My paragraph Blah, blah, and more blahs…

All other sectional tags use a "long" rendering style, in which case the section title takes a separate line on its own:

My section

Blah, blah, and more blahs…

When customizing the rendering of sectional titles, we recommend that you follow the same conventions: render paragraphs and subparagraphs in the short way and all others in the long way.

Besides their rendering, several other aspects of sectional tags can be customized:

- The call-back macro *x*-clean can be used for resetting some counters when a new section is started. For example, in order to prefix all standard environments by the section counter, you may use the following lines:

  ```
  ⟨assign|section-clean|⟨macro|⟨reset-subsection⟩⟨reset-std-env⟩⟩⟩
  ⟨assign|display-std-env|⟨macro|nr|⟨section-prefix⟩nr⟩⟩
  ```

- The call-back macro *x*-header should be used in order to modify page headers and footers when a new section is started. Typically, this macro should call header-primary or header-secondary, or do nothing.
- The call-back macro *x*-toc should be used in order to customize the way new sections appear in the table of contents.

13.7 Customizing numbered textual environments

TeXmacs provides three standard types of numbered textual environments: theorem-like environments, remark-like environments, and exercise-like environments. The first two types of environments are also called "enunciations". The following aspects of numbered textual environments can easily be customized:

- Adding new environments.
- Modifying the rendering of the environments.
- Numbering the theorems in a different way.

Defining new environments

It is possible to introduce new environments using the meta-macros new-theorem, new-remark, and new-exercise. These environments take two arguments: the name of the environment and the name which is used for its rendering. For example, you may wish to define the environment experiment by

```
⟨new-theorem|experiment|Experiment⟩
```

The text "Experiment" will automatically be translated if your document is written in a foreign language, provided that there is an entry for this word in the TeXmacs dictionaries (see the section on internationalization on page 231).

Customizing the rendering

The main rendering of numbered textual environments can be customized by redefining the macros render-enunciation, render-theorem, render-remark, and render-exercise. These macros take the *name* of the environment (like "Theorem 1.2") and its *body* as arguments. For instance, if you want theorems to appear in a slightly indented way, then you may redefine render-theorem as follows:

```
⟨assign|render-theorem|
    ⟨macro|which|body|
        ⟨padded-normal|1fn|1fn|
            ⟨surround|⟨theorem-name|which⟨theorem-sep⟩⟩||
                ⟨with|font-shape|italic|par-left|⟨plus|par-left|1.5fn⟩|body⟩⟩⟩⟩⟩
```

This redefinition produces the following effect:

THEOREM 13.2. *This is a theorem that has been indented.*

The macros render-theorem and render-remark are both based on the macro render-enunciation. The only difference between theorems and remarks is that theorems typically use an italic font. Note that proofs are rendered using render-proof; this macro is based on render-remark.

As you may have noticed by examining the above redefinition of render-theorem, it is also possible to customize the way names of theorems are printed or redefine the separator between the name and the body. Indeed, these aspects are controlled by the macros theorem-name and theorem-sep. For example, consider the following redefinitions:

```
⟨assign|theorem-name|⟨macro|name|⟨with|color|dark red|font-series|bold|font-shape|small-caps|name⟩⟩⟩
⟨assign|theorem-sep|⟨macro|: ⟩⟩
```

Then theorem-like environments will be rendered as follows:

PROPOSITION 13.3: *This proposition is rendered in is a fancy way.*

Customization of the numbering

All numbered environments such as sections and theorems come with counters *section-nr*, *theorem-nr*, etc. In order to increase, reset, or display the counter *theorem-nr*, you should use the companion macros inc-theorem, reset-theorem, and display-theorem. By redefining these macros, you may customize the way in which environments are numbered. For instance, by redefining inc-theorem, you may force theorems to reset the counter of corollaries:

```
⟨quasi|
    ⟨assign|
        inc-theorem|
        ⟨macro|⟨compound|⟨unquote|inc-theorem⟩⟩⟨reset-corollary⟩⟩⟩⟩
```

Notice the trick with quasi and unquote in order to take into account any additional action that might have been undertaken by the previous value of the macro inc-theorem. Indeed, ⟨unquote | *inc-theorem*⟩ gets replaced by the previous value of the macro inc-theorem. This macro has zero arguments and ⟨compound|⟨*unquote*|*inc-theorem*⟩⟩ simply evaluates its body.

TeXmacs organizes counters in various counter groups, which allows you to simultaneously reset all counters of a certain type at the start of a new section. The following code from the `number-long-article` style package is used in order to prefix all standard environments (theorems, equations, figures, etc.) with the number of the current section:

```
⟨assign|section-clean|⟨macro|⟨reset-subsection⟩⟨reset-std-env⟩⟩⟩
⟨assign|display-std-env|⟨macro|nr|⟨section-prefix⟩nr⟩⟩
```

13.8 Customizing list environments

Item lists and enumerations are made up of two main ingredients: the outer list environment and the inner list items. For instance, consider the following list, together with its corresponding source code:

```
⟨itemize|
    ⟨item⟩First item.
    ⟨item⟩Second item.
    ⟨item⟩Third item.⟩
```

- First item.
- Second item.
- Third item.

Then the itemize tag corresponds to the outer list environment, whereas the bullets of the inner list items are produced by the item tag.

The rendering of the outer list environment is controlled by the render-list macro, which takes the body of the list as its argument. For example, consider the following redefinition of render-list:

```
⟨assign|render-list|
    ⟨macro|body|
        ⟨surround|
            ⟨no-page-break*⟩⟨vspace*|0.5fn⟩|
            ⟨right-flush⟩⟨vspace|0.5fn⟩⟨no-indent*⟩|
            ⟨with|par-left|⟨plus|par-left|3fn⟩|par-right|⟨plus|par-right|3fn⟩|body⟩⟩⟩⟩
```

This redefinition affects the rendering of all list environments (itemize, enumerate, etc.) and reduces their right margin by `3fn`:

- This text, which has been made so long that it does not fit on a single line, is indented on the right-hand side by `3fn`.

    1. This text is indented by an additional `3fn` on the right-hand side, since it occurs inside a nested list environment.

- Once again: this text, which has been made so long that it does not fit on a single line, is indented on the right-hand side by `3fn`.

In a similar way, you may customize the rendering of list items by redefining the macros aligned-item and compact-item. Both these macros take the text for marking the item as their only argument (e.g. a bullet or a number). The macro aligned-item aligns the marker to the right (so that subsequent text is left-aligned), whereas compact-item aligns it to the left (so that subsequent text may not be aligned).

For example, consider the following redefinition of aligned-item:

```
⟨assign|aligned-item|
    ⟨macro|x|
        ⟨concat|
            ⟨vspace*|0.5fn⟩|
            ⟨with|par-first|-3fn|⟨yes-indent⟩⟩|
            ⟨resize|⟨embbb|x⟩|⟨minus|1r|2.5fn⟩||⟨plus|1r|0.5fn⟩|⟩⟩⟩⟩
```

Then items inside all list environments with aligned items will be “blackboard emboldened”:

- Items in aligned description lists are rendered using aligned-item.

    C1. First condition.

    C2. Second condition.

- Items in compact description lists are rendered using compact-item.

    Gnus and gnats. Nice beasts.

    Micros and softies. Evil beings.

In this example, we used ⟨resize|⟨embbb|x⟩|⟨minus|1r|2.5fn⟩||⟨plus|1r|0.5fn⟩|⟩ to put ⟨embbb | x⟩ inside a box of width `3fn`: the new left limit of the box is `2.5fn` to the left of the original right limit `1r`; the new right limit is `0.5fn` farther to the right. We next used ⟨with|par-first|-3fn|⟨yes-indent⟩⟩ to move the resulting box `3fn` to the left, thereby aligning it to the right[13.1].

Remark 13.4. Both aligned-item and compact-item are inline macros. They are also base macros for various other internal macros aligned-space-item, long-compact-strong-dot-item, etc. that are used for the rendering of the different types of lists (itemize-arrow, description-long, etc.).

The `std-list` package also provides a macro new-list to define new lists. Its syntax is ⟨new-list|*name*|*item-render*|*item-transform*⟩, where *name* is the name of the new list environment, *item-render* an (inline) macro for rendering the item and *item-transform* an additional transformation that is applied on the item text. For instance, the enumerate-roman environment is defined by

```
⟨new-list|enumerate-roman|aligned-dot-item|⟨macro|x|⟨number|x|roman⟩⟩⟩
```

13.1. There are simpler ways to achieve the same effect, but the way we presented here has the advantage that the alignment remains correct when boxes are moved during line breaking.

Chapter 14
Extend beyond imagination

One major feature of TeXmacs is the possibility to extend the editor using the GUILE-SCHEME *extension language*. Such extensions can be simple, like a personal boot file with frequently used keyboard shortcuts, or more complex, like a plug-in with special editing routines for documents of a particular type. The SCHEME language can also be used interactively from within the editor or invoked by special markup like “actions”.

Why SCHEME? The choice may indeed seem somewhat odd if you are accustomed to more conventional programming languages such as C++, JAVA, or PYTHON. In particular, SCHEME's heavily parenthesized syntax may scare more than one person. But it turns out that this baroque syntax has several advantages for our application. For one thing, it is easy to learn. More importantly, it enables us to treat programs and data in a uniform manner, which is particularly useful in the light of section 12.2.3. Other major advantages of SCHEME are its interactivity and high level of abstraction.

This chapter starts with a crash course on SCHEME [56]. There are several more extensive books on this topic, such as [15, 11, 52]. We also recommend to take a look at the GUILE reference manual [22], since this is the SCHEME implementation on which TeXmacs is currently based.

The rest of the chapter provides an introduction to programming TeXmacs extensions in SCHEME. More explanations on this subject can be found in Help ▸ Scheme extensions. It is also instructive to take a look at those TeXmacs source files that are written in SCHEME. These files can be found in the `progs` subdirectory of the path `$TEXMACS_PATH` where you installed TeXmacs. For instance, `$TEXMACS_PATH/progs/math/math-kbd.scm` lists the keyboard shortcuts in math mode.

14.1 A quick introduction to SCHEME

14.1.1 SCHEME sessions and atomic data types

TeXmacs uses SCHEME through an interactive interpreter. For a first encounter with the language, we invite you to launch an interactive SCHEME session using Insert ▸ Session ▸ Scheme. This allows you to enter commands after the SCHEME prompt and execute them by pressing ↩:

```
Scheme] (+ 1 1)
2
```

As you can see in this stunning example, applying a function `f` to the arguments $\mathtt{a}_1, \ldots, \mathtt{a}_n$ is done using `(f a₁ ... aₙ)`. Besides numbers, other fundamental atomic data types in SCHEME are booleans (`#t` and `#f`), characters (`#\a`, `#\b`,

`#\c`,...), strings (`"hi"`, `"there"`,...) and symbols (`x`, `fun`, `gnu2018`, `my-sym`, `x/3*`,...). The following mini-session shows a few typical computations with such objects:

```
Scheme] (> 3 2)

#t

Scheme] (if (> 2 3) "yes" "no")

"no"

Scheme] (string->list "Bonjour")

(#\B #\o #\n #\j #\o #\u #\r)

Scheme] (string-append "Hi" " " "there" "!")

"Hi there!"

Scheme] (string->symbol "hop@boeh!stuff*")

hop@boeh!stuff*

Scheme] (symbol? 'my-sym)

#t
```

One difference between strings and symbols is the notation, since strings need to be double quoted. Moreover, strings are literal constants, just like the numbers `2` and `3.14`, whereas symbols are evaluated by SCHEME. One may prevent a SCHEME expression from being evaluated using `'` as a prefix operator; this explains why we used `'my-sym` in order to obtain the symbol `my-sym`. From a computational point of view, symbols are sometimes more efficient: the equality of two symbols can be checked in unit time, whereas all characters in two strings need to be compared.

Symbols are also used as names for variables and functions. We note that SCHEME is very flexible when it comes to names of functions: all characters can be used except for whitespace and the special characters `()[]{}#.,;'"‘\`. This makes it possible to use suggestive names `string->symbol` and `symbol?` for converters and predicates. Also note that the dash `-` tends to be used (instead of `_`) as a connector for multiple-word function names such as `string-append`. In particular, many function names are prefixed by the type of their principal argument.

14.1.2 Lists and SCHEME trees

Expressions of the form (x_1 ... x_n) are also called *lists* and they are one of the most fundamental data structures in SCHEME. Lists are either constructed using `list` (from its list of entries) or `cons` (from the first *head* entry and the *tail* list of remaining entries):

```
Scheme] (list (+ 1 2 3) "hopsa" (= 2 3.4))

(6 "hopsa" #f)
```

```
Scheme] (cons (* 1 2 3 4 5) (list 6 7 8 9 10))
(120 6 7 8 9 10)
```

Notice that the second method corresponds to the way lists are stored in memory: the empty list corresponds to a special constant, whereas non-empty lists are stored as pairs with the first element of the list and a pointer to the tail. One may test for emptiness using the `null?` predicate and access the head and the tail using the `car` and `cdr` accessors. Moreover, `(caar l)`, `(cadr l)`, etc. can be used as shorthands for `(car (car l))`, `(car (cdr l))`, etc.

```
Scheme] (define l (list 1 2 3))
Scheme] (list (null? l) (car l) (cdr l) (caddr l)
              (null? (cdddr l)))
(#f 1 (2 3) 3 #t)
```

Lists are also quite convenient for the representation of TeXmacs document fragments. For instance, `(frac (sqrt "a") "b")` represents the fraction $\frac{\sqrt{a}}{b}$. One may use the prefix operator `'` to construct explicit SCHEME expressions of this kind, as we saw before in the case of symbols. The general mechanism is called *quoting* and it is extremely convenient:

```
Scheme] (sqrt "a") ;; try to apply the function sqrt to "a"
Wrong type (expecting real number): "a"
Scheme] '(sqrt "a") ;; prevent evaluation using quoting
(sqrt "a")
```

SCHEME also offers the prefix operator `` ` `` for *quasi-quoting*. Like quoting, quasi-quoting switches off evaluation; however, the evaluation of specific parts of the expression can be switched on again using the prefix operators `,` and `,@` (called *unquoting* and *unquote-splicing*):

```
Scheme] `(hop (hola ,(+ 1 2) ,(string->list "hop"))
              (hola ,(* 3 4) ,@(string->list "hop")))
(hop (hola 3 (#\h #\o #\p)) (hola 12 #\h #\o #\p))
```

Those SCHEME expressions that correspond to snippets of TeXmacs documents will be called SCHEME *trees*. A SCHEME tree is either a string or a list of the form `(l t`$_1$` ... t`$_n$`)`, where `l` is a TeXmacs primitive or macro and `t`$_1$`,...,t`$_n$ are other SCHEME trees. One may convert such SCHEME trees to actual TeXmacs trees and *vice versa* using the routines `stree->tree` and `tree->stree`:

```
Scheme] (stree->tree '(math (frac (sqrt "a") "b")))
```

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

```
Scheme] (define my-name "John")
```

```
Scheme] (stree->tree
         `(concat "My name is " (strong ,my-name) "."))
```

My name is **John**.

14.1.3 Declaring functions and macros

New SCHEME functions can be declared using `define`:

```
Scheme] (define (fibonacci n)
          (if (<= n 1) 1
              (+ (fibonacci (- n 1))
                 (fibonacci (- n 2)))))
Scheme] (fibonacci 25)
121393
Scheme] (define (.. i j)
          (if (< i j) (cons i (.. (+ i 1) j)) (list)))
Scheme] (.. 0 15)
(0 1 2 3 4 5 6 7 8 9 10 11 12 13 14)
```

One important feature of SCHEME is that it is a functional programming language: functions can both be used as arguments and return values for other functions. For example, you may use `apply` and `map` to apply a function to a list of arguments or to each element of a list:

```
Scheme] (apply + (list 1 2 3 4))
10
Scheme] (map fibonacci (.. 0 15))
(1 1 2 3 5 8 13 21 34 55 89 144 233 377 610)
```

The following syntax allows use to define functions that return functions:

```
Scheme] (define ((unary-compose f g) x) (f (g x)))
Scheme] (unary-compose fibonacci fibonacci)
#<procedure #f (x)>
Scheme] ((unary-compose fibonacci fibonacci) 5)
34
```

SCHEME also provides syntactic sugar to declare functions with an arbitrary number of arguments. The following improvement of `unary-compose` shows an example:

```
Scheme] (define ((compose f . gs) . xs)
          (apply f (map (lambda (g) (apply g xs)) gs)))
Scheme] ((compose - * +) 1 2 3 4)
          ;; (- (* 1 2 3 4) (+ 1 2 3 4)
14
```

Notice that `(lambda (g) (apply g xs))` stands for the function that takes g as input and that returns `(apply g xs)`.

Yet another powerful feature of SCHEME is the possibility to enrich the language with new primitives, through the declaration of SCHEME *macros*. This is done using the keyword `define-macro` that has a similar syntax as `define`, but with the difference that the return value is evaluated a second time.

```
Scheme] (define-macro (sum i vals . body)
          `(apply + (map (lambda (,i) ,@body) ,vals)))
Scheme] (sum k (.. 0 10) (* k k))
          ;; (apply + (map (lambda (k) (* k k)) (.. 0 10)))
285
```

This example also illustrates the power of quasi-quotation in combination with the macro system.

TEXMACS heavily relies on this possibility to extend the SCHEME language with new macros. For instance, new keyboard shortcuts and menus can be defined using the `kbd-map` and `tm-menu` macros, by means of a convenient syntax. In fact, function and macro declarations themselves should be done using `tm-define` and `tm-define-macro`: as will be detailed in section 14.3.2 below, these enhanced versions of `define` and `define-macro` make it possible to overload functions and macros in a contextual way; they also allow you to enrich functions with extra information that may be exploited by the graphical user interface, such as "suggested values" for certain function arguments.

14.1.4 Basic control structures

Local variables in SCHEME are introduced using the keyword `let*`, whereas `set!` can be used to assign a new value to such a variable:

```
Scheme] (define (foo x)
          (let* ((a (* x x))
                 (b (+ a x)))
            (set! a (* a b))
            (+ a x)))
Scheme] (foo 100)
101000100
```

The `if` primitive can be used both as an instruction and as an expression that returns a value. SCHEME programmers also frequently rely on the `cond` primitive which provides an elegant abbreviation for expressions like

```
(if c1 expr1
    (if c2 expr2
        ...))
```

For instance:

```
Scheme] (define (mix l1 l2)
          (cond ((null? l1) l2)
                ((null? l2) l1)
                (else `(,(car l1) ,(car l2)
                        ,@(mix (cdr l1) (cdr l2))))))
Scheme] (mix (list 'a 'b 'c) (list 1 2 3 4 5))
```

```
(a 1 b 2 c 3 4 5)
```

The boolean combinators `and` and `or` can take an arbitrary number of arguments. In fact, the types of the arguments are allowed to be arbitrary with the property that any value distinct from `#f` is considered to be "true". More precisely, the evaluation of `(and x1 ... xn)` stops as soon as the first x_i in the list evaluates to `#f` (in which the return value is `#f`). If this never happens, then the evaluation of x_n is returned. Similar rules apply for the `or` primitive and various other functions that manipulate boolean values.

```
Scheme] (define a #f)
Scheme] (define (set!a b) (set! a b) a)
Scheme] (list (and "hi" "there")
              (and "hi" (set!a 10) #f (set!a 100))
              a)
```

```
("there" #f 10)
```

14.2 Customizing the graphical user interface

14.2.1 Your personal boot file

So far, we have seen how to execute SCHEME commands from interactive sessions. In order the develop more permanent SCHEME code for TEXMACS, the simplest method is to create a personal boot file that should be named

```
$TEXMACS_HOME_PATH/progs/my-init-texmacs.scm
```

We recall that `$TEXMACS_HOME_PATH` usually points to the `.TeXmacs` subdirectory of your home directory ~. If it exists, then the personal boot file is assumed to contain a SCHEME program that will be executed every time you launch TEXMACS. As a first test, you may create a personal boot file with a single line

```
(display* "Hello world!\n")
```

Every time you launch TEXMACS from the command line, this should display `Hello world!` on your terminal.

14.2.2 Adding your own keyboard shortcuts

You may use the command `kbd-map` in order to define new keyboard shortcuts. For example, if you wish to be able to start a new definition by typing D e f . (and similarly for lemmas, propositions, and theorems), then it suffices to add the following code to your personal boot file:

```
(kbd-map
  ("D e f ." (make 'definition))
  ("L e m ." (make 'lemma))
  ("P r o p ." (make 'proposition))
  ("T h ." (make 'theorem)))
```

Each shortcut is of the form `(s cmd)`, where `s` is a string of key-combinations and `cmd` the SCHEME command that should be executed. The command `(make name)` starts a new tag with a given `name` at the current cursor position.

It is also possible to define shortcuts that only apply under certain circumstances. Assume for instance that we wish to define the shortcut p i for π, but only in math mode. Assume in addition that we wish to be able to quickly enter π^2 and $2\pi\mathrm{i}$ as variants. This can be achieved as follows:

```
(kbd-map
  (:mode in-math?)
  ("p i" "<mathpi>")
  ("p i var" (insert '(concat "<mathpi>" (rsup "2"))))
  ("p i var var" "2*<mathpi>*<mathi>"))
```

In this example, the command `(insert t)` is used to insert a given TeXmacs tree `t` at the current cursor position. Since the insertion of strings is very common, it is allowed to use `("p i" "<mathpi>")` as a shorthand for `("p i" (insert "<mathpi>"))`.

Special keys such as ↩, ⌫, →, etc. carry special names `return`, `backspace`, `right`, etc. Keyboard combinations using "Shift", "Control", "Alter", and "Meta" are obtained using the prefixes `S-`, `C-`, `A-`, and `M-`. In order to define ⌘^↩ as a keyboard shortcut for inserting a big vertical space, one may therefore use

```
(kbd-map
  ("C-M-return" (insert '(vspace "2fn"))))
```

Some simple keyboard shortcuts such as → or ^↩ are so heavily overloaded that TeXmacs treats them in a special way. For example, → triggers the action `(kbd-right)`, which in turn invokes the function `kbd-horizontal` with suitable arguments. Instead of redefining the → shortcut, it is therefore better to customize the function kbd-horizontal whenever appropriate. The customization of existing functions will be discussed in more detail in section 14.3.2 below.

14.2.3 Adding your own menus

$\text{T}_\text{E}\text{X}_{\text{MACS}}$ menus are generated through special SCHEME functions such as texmacs-menu, insert-menu, or texmacs-extra-menu. Simple menus are created using the macro menu-bind:

```
(menu-bind hello-menu
  (-> "Opening"
      ("Dear Sir" (insert "Dear Sir,"))
      ("Dear Madam" (insert "Dear Madam,")))
  (-> "Closing"
      ("Yours sincerely" (insert "Yours sincerely,"))
      ("Greetings" (insert "Greetings,"))))
```

Simple menu entries are of the form (label cmd), whereas -> is used in order to create submenus. The menu hello-menu needs to be attached to one of the standard menus in order to be accessible through the interface. Top-level menus for your personal use should be added to texmacs-extra-menu (so that they will appear just before the Focus menu):

```
(menu-bind texmacs-extra-menu
  (=> "Hello" (link hello-menu))
  (former))
```

The syntax => corresponds to a top-level pulldown submenu, whereas former refers to the previous value of texmacs-extra-menu. We will come back to the former primitive in section 14.3.2 below. Here it provides us with a clean mechanism to extend the menu texmacs-extra-menu more than once, if needed, possibly in different SCHEME modules.

You may use assuming or when for conditional menus; in the case of when, the conditional menu items are greyed whenever the condition is not met. In order to add hello-menu to insert-menu under the condition that we are using the letter style, you may proceed as follows:

```
(menu-bind insert-menu
  (former)
  (assuming (style-has? "letter-style")
    --- ;; horizontal separating bar
    (link hello-menu)))
```

14.3 Extending the editor

14.3.1 Manipulation of active content

All TEXMACS documents or document fragments can be thought of as *trees*. There are three data types that correspond to such trees:

Scheme trees. We have already encountered the type `stree` of *scheme trees* in section 14.1.2; for instance, the scheme tree `(frac (concat "a" (rsup "2")) "b+c")` represents the formula $\frac{a^2}{b+c}$. Scheme trees can in principle be manipulated using stand-alone SCHEME programs that do not even require TEXMACS to be installed on your computer.

TEXMACS trees. The type `tree` of *TEXMACS trees* is a data type that is hard-coded in the C++ source code of TEXMACS. It can in particular be used to represent active document fragments that are visible in TEXMACS windows. The `tree` type is an extension to the SCHEME language that is specific to TEXMACS and that is only available inside the editor.

Hybrid trees. Often, it is also convenient to mix the above types of trees into a super-type `tm` of so-called *hybrid trees*. More precisely, a hybrid tree is either a string, a tree, or a list of the form `(l` $\mathtt{x}_1$ `...` $\mathtt{x}_n$`)`, where `l` is a symbol and $\mathtt{x}_1, \ldots, \mathtt{x}_n$ are other hybrid trees.

Here is one example of each of these types of trees:

```
Scheme] (define st '(frac (concat "a" (rsup "2")) "b+c"))
Scheme] (define tt (stree->tree st))
Scheme] (define ht `(math (concat ,tt "+" ,tt)))
```

You may already have noticed that `tree` objects are pretty-printed inside SCHEME sessions:

```
Scheme] ht

(math (concat <tree <frac|a<rsup|2>|b+c>> "+" <tree
<frac|a<rsup|2>|b+c>>))

Scheme] (tm->tree ht)
```

$$\frac{a^2}{b+c}+\frac{a^2}{b+c}$$

The predicates `stree?`, `tree?`, and `tm?` can be used to test whether a given SCHEME object is a tree of type `stree`, `tree`, or `tm`:

```
Scheme] (map (lambda (p?) (map p? (list st tt ht)))
             (list stree? tree? tm?))

((#t #f #f) (#f #t #f) (#t #t #t))
```

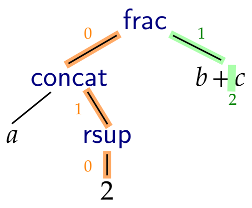
One special TeXmacs tree is the *root tree* (`root-tree`). Each open TeXmacs document is a child of the root tree and vice versa. Arbitrary subtrees of the root tree are called *active trees* and each such subtree is aware of its position inside the root tree. Using dedicated SCHEME routines from the tree API, this allows us to make changes in documents simply by assigning new values to active TeXmacs trees. In order to use these routines, one first has to import them from the appropriate module:

```
Scheme] (use-modules (utils library tree))
```

The following routine can then be implemented to clear the buffer being edited:

```
Scheme] (define (clear-document)
          (tree-set (buffer-tree) `(document "")))
```

Positions within trees are indicated through lists of numbers called *paths* and so are cursor positions. For instance, the superscript 2 in the SCHEME tree `(frac (concat "a" (rsup "2")) "b+c")` corresponds to the path `(0 1 0)`, whereas the cursor position just behind the + in the denominator corresponds to the path `(1 2)`:



The primitives `tree-set`, `tree-ref`, and `tm-ref` can be used for the modification and retrieval of subtrees along paths:

```
Scheme] (tm-ref st 0 1 0)

"2"

Scheme] (tree-set tt 0 1 0 "3")
```

3

```
Scheme] (tm->tree `(math ,ht))
```

$$\frac{a^3}{b+c}+\frac{a^3}{b+c}$$

Another useful macro for writing editing routines is `with-innermost`: it looks for the innermost tree at the current cursor position that matches a certain predicate, assigns this tree to a local variable, and then runs some code. If the predicate cannot be matched, then nothing is done. If a tag is provided instead of a predicate, then we test whether the root of the tree is labeled by that tag.

Together with what precedes, we are now in a position to define a routine that swaps the numerator and the denominator of a fraction, and attach it to the keyboard shortcut ^%:

```
Scheme] (define (fraction-swap)
          (with-innermost t 'frac
            (tree-set t `(frac ,(tm-ref t 1)
                               ,(tm-ref t 0)))))
Scheme] (kbd-map ("C-%" (fraction-swap)))
```

14.3.2 Contextual overloading

For large software projects, it is important that different modules can be developed as independently as possible. The mechanism of *contextual overloading* is of great help here: it allows you to implement a default version of a routine in a base module, and then customize this implementation in other modules.

In order to get the main idea, consider the implementation of a given functionality, like hitting the return key. Depending on the context, different actions have to be undertaken: by default, we start a new paragraph; inside a table, we start a new row; etc. A naive implementation would check all possible cases in a routine `kbd-enter` and call the appropriate routine. However, this makes it impossible to add a new case in a new module without modifying the module that defines `kbd-enter`. By contrast, the system of contextual overloading allows the user to *conditionally* redefine the routine `kbd-enter` several times in distinct modules.

Let us illustrate how this works on a simple example. Assume that we want to define a function `hello` that inserts "Hello" by default, but "hello()" in mode math, while positioning the cursor between the brackets, after the sixth character. Using contextual overloading, this can be done as follows:

```
(tm-define (hello)
  (insert "Hello"))

(tm-define (hello)
  (:mode in-math?)
  (insert-go-to "hello()" '(6)))
```

The keyword `:mode` specifies that the second declaration only applies in math-mode, when `(in-math?)` evaluates to `#t`. The order in which routines are overloaded is important. TeXmacs first tries the latest (re)definition. If this definition does not satisfy the requirements, then it tries the before-last (re)definition, and so on until an implementation is found that matches the requirements. In particular, if we swap the two declarations in the above example, then the general unconditional definition of `hello` will always prevail. If the two declarations are made inside different modules, then it is up to the user to ensure that the modules are loaded in the appropriate order.

Inside a redefinition, it remains possible to access the former definition using the keyword `former`. In particular, the code

```
(tm-define (hello)
  (if (in-math?)
      (insert-go-to "hello()" '(6))
      (former)))
```

is equivalent to the second declaration in our example.

Contextual overloading generalizes traditional overloading on types of arguments, as in C++ and various other languages. On the one hand, it becomes possible to overload on the "context", such as the cursor being inside a mathematical formula. On the other hand, you may use conditions that do not only take into account the types of function arguments, but also their values. For instance, consider a routine `my-replace` that replaces a tree `what` by another tree `by` in the current document. Then it is easy to add an optimization for the case when `what` and `by` coincide:

```
(tm-define (my-replace what by)
  default-implementation)

(tm-define (my-replace what by)
  (:require (== what by))
  (noop))
```

Besides `tm-define` and `tm-define-macro`, several other TEXMACS extensions of SCHEME support the contextual overloading mechanism. For instance, `kbd-map` and `menu-bind` support overloading on mode.

14.4 Typesetting via SCHEME

The TEXMACS stylesheet language is very convenient for the design of simple and moderately complex macros. Certain macros however may require rewritings that are easier to program in a full-fledged language like SCHEME. In this section, we give a few examples of how this can be done.

14.4.1 Hello world!

Let us start with the traditional "hello world" example. Using the TEXMACS stylesheet language, one may implement a macro hello as follows:

```
⟨assign|hello|⟨macro|name|Hello name!⟩⟩
```

The same effect can be achieved using the following external SCHEME function:

```
(tm-define (ext-hello t)
  (:secure #t)
  `(concat "Hello " ,t "!"))
```

The `:secure` directive is added in order to indicate that the function can safely be used from within the stylesheet language: you don't want to destroy your hard disk merely by opening a $\mathrm{T_EX_{MACS}}$ document. In order to use the function `ext-hello`, we still need to create a $\mathrm{T_EX_{MACS}}$ macro hello that calls this function. This is done as follows:

```
⟨assign|hello|⟨macro|body|⟨extern|ext-hello|⟨quote-arg|body⟩⟩⟩⟩
⟨drd-props|hello|arity|1|accessible|all⟩
```

The drd-props command is used in order to specify useful properties of the hello macro: the macro takes one argument and “all” arguments are “accessible” in the sense that they can be edited from within $\mathrm{T_EX_{MACS}}$ in the same way as usual macro arguments.

14.4.2 Integration into the editor

In order to play with the examples in this section, $\mathrm{T_EX_{MACS}}$ includes a special SCHEME module that can be found in `$TEXMACS_PATH/progs/utils/misc/extern-demo.scm` and a corresponding style package `extern-demo`. When selecting the style package using Document ▸ Style ▸ Add package ▸ Example ▸ extern-demo, the SCHEME module `extern-demo.scm` is loaded automatically; this is due to the following instruction at the start of the style package:

```
⟨use-module|(utils misc extern-demo)⟩
```

The header of the `extern-demo` style package also declares a variable *extern-demo-dtd* that can be used to conditionally define editing functions for this style package. First of all, we may introduce a new editing mode for the style package:

```
(texmacs-modes
  (in-extern-demo% (style-has? "extern-demo-dtd")))
```

The `%` suffix is reserved for mode declarations. The above declaration also defines the corresponding predicate `in-extern-demo?`. At a second stage, we may add a keyboard shortcut for the macro hello from the previous subsection:

```
(kbd-map
  (:mode in-extern-demo?)
  ("H i" (make 'hello)))
```

In a similar way, you may create personal style packages and SCHEME modules in the directories

```
$TEXMACS_HOME_PATH/packages
$TEXMACS_HOME_PATH/progs
```

We recall that `$TEXMACS_HOME_PATH` defaults to the `.TeXmacs` subdirectory of your home directory.

14.4.3 Highlighting subexpressions

Let us now consider a more useful example of how to extend the typesetter with functionality that is written in SCHEME. Assume that we wish to highlight all occurrences of a subexpression such as x inside a given mathematical formula. This might for instance be handy for laptop presentations.

We start with the SCHEME function `ext-highlight` that recursively traverses a given TEXMACS tree `t` and marks all subtrees that are equal to `st`:

```
(tm-define (ext-highlight t st)
  (:secure #t)
  (cond ((tm-equal? t st) `(marked ,t)) ;; t = st
        ((tree-atomic? t) t) ;; t ≠ st and t is a leaf
        (else
          (let* ((m (lambda (u) (ext-highlight u st)))
                 (l (map m (tree-children t))))
            (if (forall? tree? l) t
                `(,(tree-label t) ,@l))))))
```

The condition `(forall? tree? l)` checks whether all elements in the list `l` are TEXMACS trees. It can recursively be verified that our routine returns the original tree `t` if and only if no hits were found. This property is important because it ensures that the highlighted tree remains accessible.

Next we add the following code to the corresponding stylesheet:

```
⟨assign|highlight|⟨macro|body|sub|⟨extern|ext-highlight|⟨quote-arg|body⟩|
⟨quote-arg|sub⟩⟩⟩⟩
⟨drd-props|highlight|arity|2|accessible|0⟩
```

We consider *body* as the 0-th argument, so the drd-props command specifies that *body* is indeed accessible.

We may now use the routine to mark all variables x inside a formula:

$$\sin(x) + \frac{x-y}{\sqrt{x}}.$$

The implementation of `ext-highlight` actually contains a problem, since the x in the numerator $x-y$ did not get highlighted. Do you understand the reason behind this "bug"?[14.1]

14.4.4 Circulant matrices

For our last example, we show how to create a macro with arguments $x_1, x_2, x_3, \ldots, x_n$ that displays the corresponding circulant matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 & \ddots & x_n \\ x_n & x_1 & x_2 & x_3 & \ddots \\ \ddots & x_n & x_1 & x_2 & x_3 \\ x_3 & \ddots & x_n & x_1 & x_2 \\ x_2 & x_3 & \ddots & x_n & x_1 \end{pmatrix}$$

The SCHEME code is as follows:

```
(tm-define ((circulant-row l) i)
  (let* ((n (length l))
         (h (sublist l (- n i) n))
         (t (sublist l 0 (- n i)))
         (m (append h t))
         (r (map (lambda (x) ‘(cell ,x)) m)))
    ‘(row ,@r)))

(tm-define (ext-circulant t)
  (:secure #t)
  (let* ((l (tree-children t))
         (n (length l))
         (m (map (circulant-row l) (.. 0 n))))
    ‘(matrix (table ,@m))))
```

The corresponding TEXMACS macro circulant with an arbitrary number of arguments is created using the xmacro primitive:

```
⟨assign|circulant|⟨xmacro|l|⟨extern|ext-circulant|⟨quote-arg|l⟩⟩⟩⟩
⟨drd-props|circulant|arity|⟨tuple|repeat|1|1⟩|accessible|all⟩
```

When evaluating a circulant matrix such as ⟨circulant | a | b | c⟩, we note that the argument *l* of xmacro contains ⟨circulant | a | b | c⟩ itself. The drd-props command specifies that a circulant matrix has $1+1\cdot n$ accessible arguments for some $n \in \{0,1,2,\ldots\}$.

14.1. Answer: this is due to the fact that x is a *substring* of $x+y$, but not a *subtree*. In order to correct the bug, we would need to add extra cases in the routine `ext-highlight` that also treat occurrences of `st` as a substring of `t`.

BIBLIOGRAPHY

[1] O. Arsac, S. Dalmas, and M. Gaëtano. The design of a customizable component to display and edit formulas. In *ACM Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation, July 28–31*, pages 283–290. 1999.

[2] B. Beeton, A. Freytag, and M. Sargent III, Unicode support for mathematics, `http://www.unicode.org/reports/tr25/tr25-13.pdf`, (2012).

[3] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, Extensible markup language (XML) 1.0 (fifth edition), `http://www.w3.org/TR/xml/`, (2008).

[4] R. Bringhurst. *The Elements of Typographic Style*. Harley and Marks, 4 edition, 2012.

[5] *The Chicago Manual of Style*. University of Chicago Press, 17th edition, 2017.

[6] CollabNet, Subversion, `https://subversion.apache.org/`, (2001).

[7] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato. *Version control with Subversion*. O'Reilly, 2nd edition, 2009.

[8] J. Cristy et al, ImageMagick, `https://imagemagick.org/`, (1990).

[9] Dafont, `https://www.dafont.com/fr/`, (2000).

[10] E. Dahlström, P. Dengler, A. Grasso, C. Lilley, C. McCormack, D. Schepers, and J. Watt, Scalable Vector Graphics (SVG) 1.1 (second edition), `http://www.w3.org/TR/SVG/`, (2011).

[11] R. K. Dybvig. *The SCHEME programming language*. Mit Press, 2009.

[12] H. Fernandes, Gnu Dr. Geo, `http://drgeo.eu/`, (1996).

[13] The Document Foundation, LibreOffice, `http://www.libreoffice.org`, (2013).

[14] H. W. Fowler. *A Dictionary of Modern English Usage*. Oxford University Press, 1926.

[15] D. P. Friedman and M. Felleisen. *The Little Schemer*. MIT Press, 1996.

[16] M. Goossens, F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1993.

[17] M. Grevisse. *Le Bon Usage*. De Boeck Supérieur (groupe Albin Michel), 16th edition, 2016.

[18] A. G. Grozin. TeXmacs interfaces to Maxima, MuPAD and Reduce. In V. P. Gerdt, editor, *Proc. Int. Workshop Computer algebra and its application to physics*, number 11-2001-279 in JINR E5, page 149. Dubna, June 2001. Arxiv cs.SC/0107036.

[19] A. G. Grozin, TeXmacs-Maxima interface, Arxiv cs.SC/0506226, (June 2005).

[20] A. G. Grozin. TeXmacs-Reduce interface. *CoRR*, abs/1204.3020, 2012.

[21] M. Gubinelli, J. van der Hoeven, F. Poulain, and D. Raux. GNU TeXmacs: towards a scientific office suite. In *Mathematical Software - ICMS 2014 - 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings*, pages 562–569. 2014.

[22] Guile reference manual, `https://www.gnu.org/software/guile/docs/docs-1.8/guile-ref/`, (2010).

[23] D. Harvey and J. van der Hoeven. Integer multiplication in time $O(n \log n)$. Technical Report, HAL, 2019. `http://hal.archives-ouvertes.fr/hal-02070778`.

[24] J. van der Hoeven. GNU TeXmacs: a free, structured, wysiwyg and technical text editor. In Daniel Filipo, editor, *Le document au XXI-ième siècle*, volume 39–40, pages 39–50. Metz, 14–17 mai 2001. Actes du congrès GUTenberg.

[25] J. van der Hoeven. *GNU TeXmacs User Manual*. HAL, 2013. `http://hal.archives-ouvertes.fr/hal-00785535`.

[26] J. van der Hoeven. Towards semantic mathematical editing. *JSC*, 71:1–46, 2015.

[27] J. van der Hoeven. Mathematical font art. In Gert-Martin Greuel, Thorsten Koch, Peter Paule, and Andrew Sommese, editors, *Mathematical Software – ICMS 2016: 5th International Conference, Berlin, Germany, July 11-14, 2016, Proceedings*, pages 522–529. Cham, 2016. Springer International Publishing.

[28] J. van der Hoeven, A. Grozin, M. Gubinelli, G. Lecerf, F. Poulain, and D. Raux. GNU TeXmacs: a scientific editing platform. *ACM Commun. Comput. Algebra*, 47(1/2):59–61, 2013.

[29] J. van der Hoeven, G. Lecerf, and D. Raux. Preserving syntactic correctness while editing mathematical formulas. In I. Kotsireas and E. Martínez-Moro, editors, *Proc. Applications of Computer Algebra 2015*, volume 198 of *Springer Proceedings in Mathematics and Statistics*, pages 459–471. Cham, 2015. Springer.

[30] M. Hohenwarter et al, GeoGebra, `http://www.geogebra.org`, (2001).

[31] R. Huddleston and G. K. Pullum. *The Cambridge Grammar of the English Language*. Cambridge University Press, 2002.

[32] B. Jackowski, J. Nowacki, and J. Ludwichowski, The TEX Gyre collection of fonts, `http://www.gust.org.pl/projects/e-foundry/tex-gyre/`, (2006).

[33] A. Jaffer, T. Lord, M. Bader et al, GNU Guile, `http://gnu.org/software/guile`, (1993).

[34] G. Kahana, Hummus, `https://pdfhummus.com`, (2011).

[35] S. Kimball, P. Mattis et al, Gimp, the GNU Image Manipulation Program, `https://www.gimp.org/`, (1996).

[36] D. E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, 1984.

[37] D. E. Knuth. *Computer Modern Typefaces*, volume E of *Computers and Typesetting*. Addison-Wesley, 1986.

[38] L. Lamport. *LaTeX, a document preparation system*. Addison Wesley, 1994.

[39] A. Larsson et al, Dia, a drawing program, `https://wiki.gnome.org/Apps/Dia`, (1998).

[40] A. Lawson. *Anatomy of a Typeface*. Hamish Hamilton, 1990.

[41] W3C M. Smith, HTML: the markup language (an HTML language reference), `http://www.w3.org/TR/html-markup/`, (2012).

[42] Microsoft, Office, `https://www.office.com/`, (1990).

[43] O. Patashnik. BibTeXing. Documentation for general BibTeX users, 1988.

[44] S. Pinker. *The sense of style*. Penguin, 2014.

[45] M. Pool et al, GNU Bazaar, `http://bazaar.canonical.com/`, (2005).

[46] G. K. Pullum. Punctuation and human freedom. *Natural Language & Linguistic Theory*, pages 419–425, 1984.

[47] Qt, `https://www.qt.io/`, (1995).

[48] J. Richter-Gebert, H. Crapo, and U. Kortenkamp, Cinderella.2, The Interactive Geometry Software, `http://cinderella.de`, (2013).

[49] D. Roundy et al, Darcs, `http://darcs.net/`, (2003).

[50] Roy Rosenzweig Center for History and New Media, Zotero, `http://www.zotero.org`, (2013).

[51] R. E. Silverman. *Git pocket guide*. O'Reilly, 2013.

[52] M. Sperber, R. K. Dybvig, M. Flatt, A. Van Straaten, R. Findler, and J. Matthews. Revised 6 report on the algorithmic language Scheme. *Journal of Functional Programming*, 19(S1):1–301, 2009. `http://www.r6rs.org/final/r6rs.pdf`.

[53] R. Stallman et al, GNU Emacs, `https://www.gnu.org/software/emacs/`, (1985).

[54] STI Pub companies, STIX fonts project, `http://www.stixfonts.org/`, (2010).

[55] W. Strunk, Jr. and E. B. White. *The Elements of Style*. Macmillan, Third edition, 1979.

[56] G. J. Sussman and Jr. G. L. Steele, Scheme, `http://www.schemers.org/`, (1975).

[57] The Bibdesk Team, Bibdesk, `http://bibdesk.sourceforge.net/`, (2002).

[58] The Inkscape Team, Inkscape, `http://www.inkscape.org`, (2003).

[59] Han Thê Thanh. *Micro-typographic extensions to the TEX typesetting system*. PhD thesis, Masaryk University Brno, 2000.

[60] F.-N. Thomas and M. Turner. *Clear and Simple as the Truth: Writing Classic Prose*. Princeton University Press, 1996.

[61] Toptal designers, Subtle patterns, `https://www.toptal.com/designers/subtlepatterns/`, (2011).

[62] L. Torvalds et al, Git, `https://git-scm.com/`, (2006).

[63] D. Turner, R. Wilhelm, W. Lemberg et al, Freetype, `https://www.freetype.org/`, (1996).

[64] Unicode Consortium, Unicode, `http://www.unicode.org/`, (1991).

[65] Wikimedia Foundation, Wikimedia, `https://www.wikimedia.org`, (2003).

[66] J. M. Williams. *Toward clarity and grace*. The University of Chicago Press, 1990.

Index