Math∫pad

Ergonomic Document Preparation

Roland Backhouse and Richard Verhoeven

Version 0.60

Roland Backhouse and Richard Verhoeven

Department of Mathematics and Computing Science,
Eindhoven University of Technology,
P.O. Box 513,
5600 MB Eindhoven,
The Netherlands.

email: mathspad@win.tue.nl
world-wide web: http://www.win.tue.nl/win/cs/wp/mathspad

Preface

Once upon a time, in the days before computers became widely used, authors of books and articles would produce a typescript of their work; before going to the printing office where the typescript was set in lead, a "copy-editor" would mark the typescript with all sorts of hieroglyphics. The hieroglyphics acted as a secret code between the editor and the printer (a human being, by the way) and served to tell the printer such things as when to use particular fonts (for example bold characters) and where to position text on the page. Mathematical texts were regarded as "difficult copy" and incurred a financial penalty.

Copy-editors still do exist, although for mathematical texts at least their function is rapidly being usurped by the authors themselves. The hieroglyphics used by the copy-editors are also being replaced by the so-called "mark-up" languages like TEX and LATEX, sgml and html[1]. Moreover, the printer of today is a computer program that automatically interprets the commands of the mark-up language.

The advantage of the old system was a clear delineation of professional responsibilities. The author was responsible for the content, the copy-editor for a professional appearance. Computerised systems tend to confuse the two tasks. Authors are now responsible for both appearance and content and, if one is not careful, paying attention to the former can have a detrimental effect on the latter. Authors no longer pay a financial penalty for difficult mathematical copy; they pay the penalty instead in their own time and energy.

The advantage of computerised systems, on the other hand, is improved quality; above all, the elimination of typographical errors introduced by the printer in the process of typesetting is a godsend for authors of papers containing substantial amounts of mathematics. This one advantage is so significant that very few authors would wish to return to the old system.

Nowadays, a battle is raging between the "WYSIWYG" ("What You See Is What You Get") and the "logical design" systems. WYSIWYG systems are dismissed as "what you see is ALL you get" by their opponents, to which the retort is that logical design systems may be "wonderful" but they are also "awful" and their users are "masochists" [6]. The essence of the dispute is that WYSIWYG systems tend to be easier to use (and to learn to use) but they are less flexible, leading to a lower quality finished product, and they can encourage authors to waste time on appearance to the detriment of the

[1]The abbreviations, "sgml" and "html" stand for standard generalized markup language and hypertext markup language, respectively.

content.

Math∫pad combines the advantages of WYSIWYG and logical design systems in a way that harks back to the old-fashioned manual system. It is first and foremost a logical design system —logical elements within a Math∫pad document are defined by "templates" and manipulated by a structure editor. Unlike conventional logical design systems, however, it is a "quasi"-WYSIWYG system in the sense that the readability of mathematics displayed on screen is as near to the readability of mathematics in a printed book as current technology will permit. Moreover, Math∫pad is not a "what-you-see-is-ALL-you-get" system. Nor is it a system that is exclusively for mathematicians — even if the documents you write contain no mathematics whatsoever, its unique template system can lead to a dramatic improvement in your productivity.

The essence of Math∫pad is that a document is a structure on which two "views" are defined. One "view" is what is displayed on screen; as mentioned earlier, in this view, the mathematics is as near as possible to the form desired by the user as current technology permits. Otherwise the view resembles the galley proofs used in the good old-fashioned manual type setting systems: the document is not split into pages, equations are not numbered but named, the position of figures and tables is not fixed, and so on. The second view is a sequence of commands written in a mark-up language like LaTeX. This second view, generated automatically when a command to "output" the document is given, is used to create the final document. The first view is what concerns the author, the second view is what concerns the copy-editor.

An author using Math∫pad will begin by writing his/her article, using conventional mathematical notation, concentrating on the content and ignoring all issues of layout, page breaks, centering etc. Subsequently, "copy-editing" instructions can be added to the document instructing the "printer" (now software rather than a human being) how to produce the finished document. For draft versions of an article or book the instructions can easily be added by the author himself but for the final version (particularly in the case of a book) it may be necessary and will often be desirable to call on the services of a professional copy editor (now titled a "LaTeXpert", perhaps?).

A Math∫pad document ready for processing will thus resemble an old-fashioned copy-edited text on its way to the printing department. Strewn among the author's text there will be a number of "commands" instructing the printer how to set the text — the style of headings and page numbering to be used, where page breaks are desired etc. Unlike raw mark-up code, however, these commands do not dominate the text; since they tend to be congregated in certain places, for example in the so-called preamble, the author's text remains eminently readable. (Using raw LaTeX, for example, the situation is quite different since the code needed to typeset mathematics tends to swamp everything else.)

The MathSpad system represents a radical departure from existing design philosophies making it difficult to describe to newcomers. The best way to find out what it offers is to just try it and see. Installment of the system (given the right platform) is just a single command and then you are away[2]. So, why not go out and give it a whirl!?

* * *

Development of the MathSpad system was initiated in 1987 by Roland Backhouse with support from the University of Groningen in the Netherlands and, later, the NWO (the Dutch organisation for scientific research). In intensive discussions with Paul Chisholm a design was formulated of a system for on-screen algebraic calculations. This prototype system, implemented by Paul with technical support from Harm Paas, first became available for use in 1990. The principal design element was a flexible system for on-the-fly definition of mathematical notations (the "stencils" of the current system).

During 1990 much was learnt from experimenting with Paul's prototype implementation. "Guinea pigs" at that time who provided much useful feedback were Grant Malcolm and Ed Voermans. During a period of six months several alterations were made to the ergonomics of the system as a result of this feedback. In the summer of 1990 Roland Backhouse moved to Eindhoven University of Technology and Paul Chisholm left the university world for a post in industry in Australia. The system had settled into a stable state and it was decided to freeze the development for a while until full benefit had been gained from its active use. Ed Voermans was responsible for correcting the few bugs that emerged after Paul's departure. Being a prototype, the system had at that time no name. A paper written by Paul Chisholm outlining the system was presented in Paul's absence by Roland Backhouse at the 3rd International Workshop on Software Engineering and its Applications, Toulouse, France, 3–7 December 1990.

Work on MathSpad itself began in March 1992. Richard Verhoeven and Olaf Weber took on the task of a complete reimplementation as their final year project (in Dutch "afstudeerproject"). Priorities were to increase the (on-screen) readability of documents by the use of proportional spacing, the integration of plain text with mathematical

[2]The system can be downloaded from the URL given on the copyright page; the README file you find there gives full details of the software you need to install the system. The system is also available from a number of different archiving services.

calculations, and a better, more flexible design of stencils. The transformation system that formed a major part of the existing prototype was scrapped because, although the prototype system *had* been intensively used on several projects, this element *had not* — possibly due to bad design but more likely due to the inherent instability of the mathematical frameworks being investigated by those using the system.

Richard and Olaf graduated in December 1992 and were then employed for a period of three months to refine their implementation. After this period Olaf left and has since joined a similar project at the CWI in Amsterdam. Richard Verhoeven is currently employed at Eindhoven University of Technology and is responsible for the further development of Math∫pad. Ed Voermans suggested the name. The Math∫pad logo was designed by Q Design from an original idea by Kevin Backhouse; the design was subsequently implemented in METAFONT by Kevin.

During 1993, Math∫pad was used intensively by members of the Mathematics of Program Construction group at Eindhoven University of Technology and, once again, the feedback obtained was extremely valuable in improving the system's design. Version 0.31 of the system was made available via ftp in December, 1993.

Further use of the system after this initial release led to a major revision in the way files are stored, and to the inclusion of extensive help facilities. These revisions were designed principally to accommodate co-authorship at a distance.

We continue to welcome criticism and suggestions on the design.

We are extremely grateful to all those who have been willing to act as guinea pigs during periods when Math∫pad was both unstable and sometimes unreliable — Eerke Boiten, Henk Doornbos, Maarten Fokkinga, Netty van Gasteren, Rik van Geldrop, Paul Hoogendijk, Ed Knapen, Ed Voermans, Harold Weffers and Jaap van der Woude. Hilary Backhouse and Maarten Fokkinga provided us with many detailed comments on, criticisms of, and corrections to, the first draft of this manual. Lambert Meertens has also offered sage and much appreciated advice from the very earliest days of the project.

Our biggest thanks go to Hilary Backhouse for her many contributions which include preparing the index and appendix, numerous suggestions for improving our English, and for carrying out the task of incorporating a great many alterations to the text. Of course, the responsibility for any errors in the text is entirely ours.

Roland Backhouse	Eindhoven
Richard Verhoeven	December, 1995.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Welcome

Welcome to Math∫pad. This introductory chapter has been written to help you assess whether Math∫pad could be a useful tool to help you in your work. It may be that you are already reading this document at your computer terminal. If so, then all well and good. If not, i.e. you are reading a printed copy of the manual, it is recommended that you do not read too far before obtaining a copy of the system and continuing your reading on-line. We promise that your reading pleasure will not be severely impaired in so doing and you will get a much more realistic impression of the system's ergonomics. Instructions for installing the system can be found in the README file of the distribution. If the system has already been installed at your site just type the command

```
mathspad manual &
```

and this document will appear on your screen[1]. You may first wish to reposition and/or resize the window (in the normal way) to your own liking following which you can use the scroll bars at the side of the document to move through it.

Math∫pad is a system for preparing documents (books, articles etc.) on a computer. The main strength of the system lies in the preparation of documents of a mathematical nature, particularly in the case that the mathematics is non-standard. It builds upon the power of the TeX and LaTeX document preparation systems[2] by providing a mechanism for converting documents prepared with the aid of the system into LaTeX

[1] If that doesn't work try `mathpad`, `Mathpad` or `MathSpad` instead of `mathspad`. If none of these work consult your system manager.

[2] See [3] and [5].

documents. The mechanism is extremely flexible in a very simple way so that Math∫pad can equally well be used to create troff documents or html documents, or, indeed, documents written according to any ascii-based mark-up language.

Like TEX and LATEX, Math∫pad emphasises *logical design* rather than *visual design* and is not a WYSIWYG (what you see is what you get) system. Unlike TEX and LATEX, however, Math∫pad provides an ergonomically-designed user interface that allows you to create and manipulate mathematical formulae at your computer terminal in a form whose readability approaches the high standard of graphics-based WYSIWYG systems.

Math∫pad is of potential value to you if what you write typically contains lots of items of a mathematical nature, for example formulae like:

$$\sum_{0 \leq k < m} \left\lfloor \frac{nk+x}{m} \right\rfloor \;=\; d\left\lfloor \frac{x}{d} \right\rfloor + \frac{m-1}{2}n + \frac{d-m}{2}$$

or computer programs (with or without comments of a mathematical nature):

$$\begin{array}{l}
\textbf{program}\ \ \textit{example}\,; \\
\textbf{var} \\
\quad k\,,m\ :\ \mathsf{integer}\ ; \\
\quad x\,,y\ :\ \mathsf{real}\ ; \\
\textbf{begin} \\
\quad \{\ 0 \leq m\,\} \\
\quad k\ :=\ m\,;y\ :=\ 1\,; \\
\quad \{\ \textbf{Invariant}:\ \ y{*}x^k\ =\ x^m \\
\qquad \textbf{Variant}:\ \ k\,\} \\
\quad \textbf{while}\ \ k <> 0\ \ \textbf{do begin} \\
\qquad k\ :=\ k{-}1\ \ ;\ y\ :=\ y{*}x \\
\quad \textbf{end} \\
\quad \{\ y = x^m\,\} \\
\textbf{end.}
\end{array}$$

If, however, you restrict yourself to standard mathematical notation such as taught up to, but not including, university level you will probably find that a software package that has been tailor-made to respect such standards is more suited to your purposes. For instance, you will notice —if you are reading this document on screen— that in the mathematical formula above the parentheses $\lfloor$ and $\rfloor$ (a standard notation for the floor function mapping reals to integers) do not increase in size to accommodate the size of the argument. Math∫pad has no built-in "knowledge" of accepted notational conventions. This means that on-screen formulae may not be so readable as those produced by specialised systems.

Math∫pad begins to be an attractive alternative when you wish to deviate from accepted notation and invent your own. You may prefer, for example, to write the tautology

$$((p{\Rightarrow}q){\Rightarrow}r){\Rightarrow}(p{\Rightarrow}q{\Rightarrow}r)$$

using a two-dimensional turnstile notation as shown below:

$$\begin{array}{|l} \begin{array}{|c} p \vdash q \\ \hline r \end{array} \\ \hline \begin{array}{|c} p \\ \hline q \vdash r \end{array} \end{array}$$

The most important component of the Math∫pad system is a "template"- creation mechanism which allows you to design and use your own notation — if necessary on the fly during the preparation of a document. Included in this mechanism is a method for specifying the output that your notation should generate thus ensuring a high-quality printed version of your document even if the screen version has some shortcomings.

Built into the system is an automatic mechanism for generating LaTeX output. In principle, it is thus the case that the user need have no knowledge of LaTeX to use the system. In practice, knowledge of LaTeX (or TeX) is required in order to exercise one's right to override the decisions taken by the system. This ability to override automatically generated output also enables one to create output suited to other systems. It is possible, for example, to generate *from the same* Math∫pad *document* both LaTeX and html output (for use with internet).

The use of Math∫pad truly becomes worthwhile if the documents one writes often contain symbolic calculations. Symbolic calculations are very much like the weather: occasionally sudden changes take place, but most of the time the changes are gradual. Below is an example of such a symbolic calculation. Unless you belong to a particular group you will not have any idea what it is about. This is an advantage since you will then be able to view the calculation from a purely syntactic perspective. What you will observe is that there is only one sudden change in the calculation, that expressed in the first equation. Following that the calculation consists of various rearrangements of a small number of patterns. You will note, for example, that the pattern "$f\circ$" is repeated without change, and that the subexpression "$(\![GF;\ G.f]\!)$" is also copied repeatedly but in different positions in the formula.

$$
\begin{array}{ll}
 & \beta \circ \alpha \\
= & \{ \quad \text{definitions of } \beta \text{ and } \alpha \quad \} \\
 & f \circ F.(\![GF;\ G.f]\!) \circ F.\mathsf{in}_{GF} \\
= & \{ \quad F \text{ is a functor} \quad \} \\
 & f \circ F.((\![GF;\ G.f]\!) \circ \mathsf{in}_{GF}) \\
= & \{ \quad \text{SELF} \quad \} \\
 & f \circ F.(G.f \circ G.F.(\![GF;\ G.f]\!)) \\
= & \{ \quad G \text{ is a functor} \quad \} \\
 & f \circ F.G.(f \circ F.(\![GF;\ G.f]\!)) \\
= & \{ \quad \text{definition of } \beta \quad \} \\
 & f \circ F.G.\beta \ .
\end{array}
$$

Math∫pad is in fact just a tree editor, the structure of the trees one manipulates being defined by the user. Its strength lies in the flexibility with which one can define the logical structure of formulae and how they are to be laid out on the screen, and in the ease with which one can select, copy and rearrange subexpressions in a mathematical formula.

In summary, Math∫pad has been designed as a hybrid between WYSIWYG and macro-based document preparation systems. It will appeal to you if your work demands the comprehensiveness, power and flexibility of a logical-design system like TEX or LATEX. If you choose to use Math∫pad you will not be sacrificing any of these advantages. (Nor will you have to rewrite your old documents: a Math∫pad document can include any amount of raw TEX or LATEX code. No special conversion program is necessary.) On the other hand, you will gain the indispensable advantage of a visual design system so that you can read what you have written as you write.

Math∫pad will not fulfil all your dreams. It was developed as a research project in the design of computer interfaces for the writing of mathematical documents. As such its capabilities for editing and viewing *plain* text are not so comprehensive as conventional editing systems. Even for the input of mathematics its ergonomics is hampered by the inhuman nature of keyboard-mouse input. Math∫pad is nevertheless intended as a practical, pleasant-to-use system, as judged within the context of current technology, and one that will have a place in the emerging generation of computer interfaces.

If you would like to know more about the philosophy behind the design of Math∫pad then you may continue by reading the chapter entitled "The Math∫pad Philosophy". To do so click on Load in the top-left corner of this window (in other words move the cursor so that it is pointing to the word Load and then press and release any one of your mouse buttons). Then select `Philosophy.mpd` in the file selector window that appears. (Do so by pointing the mouse at the entry `Philosophy.mpd`; then double-click with any of the mouse buttons, or click once and then either click on the OK button or press on the Enter key on your keyboard.) The contents of this window will then be replaced by that chapter and you can continue reading.

The tutorial chapter is a cross between an overview of the main elements of the system and a short tutorial on its use. To begin reading it, follow the instructions given above for loading the "Philosophy" chapter replacing the latter everywhere by "Tutorial".

If you do not want to leave this chapter just yet you can bring up the tutorial chapter alongside this one. Begin by clicking on Window in the Math∫pad console at the top of your screen. A window similar to this one will appear and then you can follow the instructions above, but, instead of beginning by clicking on Load in the top-left corner of *this* window you should click on Load in the top-left corner of the new window. Subsequently, you can quit this welcoming chapter by clicking on Done or by using the normal window-manager routine.

Finally, if you just want to go it alone then there is a context-sensitive help facility. Position the mouse anywhere in one of the Math∫pad windows and press the "Help" button on your keyboard. Part of the reference section of this manual will automatically appear on your screen and, at the head of the document, you will find some information related to the subwindow at which the mouse is pointing. From there you can scroll through the rest of the document or chase the references in order to find the information you need. (We recommend that you do read the tutorial at some stage. Otherwise, you will never fully appreciate all the possibilities that the system offers. If your keyboard doesn't have a "Help" key then you will have to consult chapter 14 or your local expert to see how to bind another key to the help function.)

Chapter 2

The Math∫pad Philosophy

OK. Let's say it up front. This is the chapter that you *certainly* don't *have* to read if you don't want to. It won't help you to use Math∫pad more effectively; nor is it likely to change your view on the suitability of Math∫pad to your particular circumstances. The chapter is purely and simply a chance for us to expound our views on some current trends in system design. Having created the Math∫pad system we feel we have a right to comment!

2.1 Needs and Priorities

Engineering design is about creating artefacts that fulfil a need to the best of the ability of current technology. Good design involves insightful decisions on what is important, and what should be given priority.

The *need* that Math∫pad attempts to fulfil is the ability to write mathematical documents, of publication quality, ergonomically. Math∫pad builds on the TEX and LATEX systems because these systems fulfil part of that need: they enable one to *produce* publication-quality documents. In fact, in this respect they are superb. So far as we are aware, the (typographical) quality of documents produced using these systems is unsurpassed at this point in time. But let's be forthright. TEX and LATEX are based on teletype technology and are a pain to use. In our view, *writing* mathematics with TEX or LATEX is like editing a program by splicing a paper tape (the way you had to do it some thirty years ago). In the words of Leonhard and Simon [6] TEX and LATEX are both "wonderful" and "awful" and their users are "masochists". Thirty years from now noone will be writing mathematics using LATEX or indeed any ascii-based markup language. Let's hope however that that situation arrives a bit sooner than after thirty years.

Why do scientists feel the need to write publication-quality documents? After all, in bygone days, getting a document to publication standard was the task of a professional copy-editor. The scientist's responsibility was getting the content right.

One answer to this question is the desire of all scientists to communicate the results of their effort in the best possible light. Such a desire we can all appreciate, but this is not the right answer. The overwhelming consideration for the scientist is the desire to eliminate the possibility of *introducing error*[1]. Accordingly, avoiding the introduction of error has been the driving force behind all design decisions in the development of MathSpad.

2.2 Avoiding the Introduction of Error

Avoiding the *introduction* of error is a much more modest aim than the *elimination* of error. We believe, however, that the former is a realistic aim for a software tool whereas the latter is not. Recognition of the difference between these two goals leads to a fundamental difference in one's approach to tool building.

MathSpad was originally developed to support the work of a research group active in the area of the mathematics of (computer) program construction. The nature of the group's work was (and is) such that their articles frequently involve substantial algebraic calculations in advanced areas of mathematics. The primary tools used by the group to carry out their work were (and still are) paper and pencil.

Performing algebraic calculations at a computer terminal and doing them by hand with paper and pencil are quite different sorts of activities. In designing MathSpad we started out from the viewpoint that the latter (using paper and pencil) is much more reliable than the former (using a computer). This is quite contrary to prevailing claims made about formal verification systems and the like. One must recognise, however,

[1]Avoiding the introduction of error as the number one priority can lead scientists to do rather strange things. In the seventies, for example, it was quite common for computing scientists to produce their books on a computer and deliver "camera-ready" copy rather than put their trust in a professional printer. The results were awful. It was obvious which books had been produced by computer and which not, and the former were a real pain to read no matter how good the content! (See the frank admission by the author of one such book —David Gries— in [2].) In those days both the reader and the writer were expected to be "masochists"; at the time that development of MathSpad began it was only the writer. Our goal was to relieve the writer's misery too.

that, for us as human beings, paper and pencil is so much *handier*; the computer, in comparison, is a very clumsy instrument, and the use of a clumsy or badly-designed tool will inevitably introduce errors of its own kind.

The traditional argument for the use of a computer for algebraic calculations, that computers are able to faithfully execute formal manipulation rules, is one that we find difficult to justify in the context of our own research. The problem is that it is frequently hard to explain to a dumb machine how a rule should be applied in a particular situation because there are —from the trained scientist's point of view— so many trivial intermediate steps that also have to be explained. The icon that is used for Math∫pad illustrates this point. Given functions f, g, θ_1 and θ_2 such that θ_1 and θ_2 have inverses θ_1^{-1} and θ_2^{-1}, respectively, we have the (straightforward) identity,

$$\theta_1 \circ f \;=\; g \circ \theta_2 \quad\equiv\quad f \circ \theta_2^{-1} \;=\; \theta_1^{-1} \circ g \quad .$$

(Here "$\circ$" denotes composition of functions. The Math∫pad icon is slightly less general, since the subscripts on θ_1 and θ_2 are omitted.) Two examples of this identity are provided by the logarithm and exponential functions. The functions are inverses of each other, and, as is well known,

$$\mathsf{exp}(-x) \;=\; \frac{1}{\mathsf{exp}(x)} \qquad \text{whereas} \qquad -\mathsf{ln}(x) \;=\; \mathsf{ln}(\frac{1}{x}) \quad ,$$

and

$$\mathsf{exp}(x{+}y) \;=\; \mathsf{exp}(x){\cdot}\mathsf{exp}(y) \qquad \text{whereas} \qquad \mathsf{ln}(x){+}\mathsf{ln}(y) \;=\; \mathsf{ln}(x{\cdot}y) \quad .$$

In order to see the correspondence between the general property and the two specific instances one must recognise that the general property has been expressed in terms of functions alone, whilst the instances are in terms of functions applied to their arguments. One must also recognise that plus and times are both *binary* functions and that this must be taken into account in an explanation that the identity

$$\mathsf{exp}(x{+}y) \;=\; \mathsf{exp}(x){\cdot}\mathsf{exp}(y) \quad\equiv\quad \mathsf{ln}(x){+}\mathsf{ln}(y) \;=\; \mathsf{ln}(x{\cdot}y)$$

is indeed an instance of the general property. This can be difficult to explain to a fellow human being but it is nothing compared to the overhead that is typically required in order to command a computer program to apply the general rule.

The working scientist (or engineer) has a need to be able to get on with productive work and requires a tool that will allow him to communicate the justification for his calculation steps at the level of formality of his own choosing. The working scientist is very conscious of the possibility of error and will choose his tools carefully in order to avoid the danger of its introduction; the responsibility for *eliminating* error, however, remains his and is not delegated either to a machine or to a colleague. It is such scientists that Mathſpad aims to cater for.

2.3 Four Priorities

Having identified the goals of the Mathſpad system, we now turn to the priorities we set in order to achieve those goals.

The first priority was *readability*. Mathematical notation has been designed over centuries to be eminently readable, so a first requirement of the Mathſpad system was that one should be able to work directly with mathematical notation (and not some approximation to mathematical notation).

Of course, we can all train ourselves to read the unreadable. Blind people learn to read braille and are able to cope admirably well. Those who have been programming for a quarter of a century or more will remember the time when it was necessary to be able to read paper tape in order to amend or correct a program. It is a skill that is soon learnt and, after some practice, it is possible to convince oneself that it can be done without thinking. To do so is however pure self-deception. Indeed, it is quite probably the case that the use of paper tape as input medium contributed far more in those days to the unreliability of computer software than did the lack of structured programming techniques.

Monospaced text, particularly when used for mathematical work, is also unreadable, although it has been around for so long that many ardent computer users do not seem to realise the training that has to be undergone to learn how to read it. To illustrate the point consider the expression `a+b.c+d` as written here in typewriter font. The justification for the mathematical convention that multiplication is denoted by the smallest symbol (a dot) and has the highest precedence is totally negated by the white space around the symbol created by the use of a monospaced font. The eye naturally groups the expression into the two subexpressions `a+b` and `c+d`, and the intended structure of the expression is completely disguised. In general, the use of a monospaced font completely frustrates attempts to impart structure to the reader. Matters are made quite intolerable when this is combined with the limitations of the ascii character set (as happens in programming languages). It is indeed our contention that the continuing use of monospaced ascii as primary input medium to computer programs is the most significant single source of programming error. The belief that one can do creative mathematics in ascii is as ridiculous as the belief that one can do creative mathematics in paper tape format.

Our second priority was *flexibility*. Math∫pad was designed to support the work of a research group engaged in the exploration of novel mathematical theories and so flexibility was a must. But flexibility is also vital to all scientists. Even if one is —for example— designing an algorithm for some well-defined, specialist task it is likely that one needs to appeal to a whole host of existing mathematical theories. (It is often the case that it is only when one submits one's design to a completely formal proof that one becomes truly aware of just how many.) The use of a system designed to support only a subset of available mathematical knowledge will be more effort than it is worth.

Flexibility means that the responsibility for error lies fairly and squarely on the shoulders of the user. All formal proof assistants of which we are aware attempt to relieve the user of some of this responsibility by restricting the available possibilities. One often sees claims like: "the system guarantees that only type-correct statements can be entered". Our view is that this is a totally misguided policy. The discipline of thoroughness must remain an integral part of the scientific ethic and not be imposed by a machine. We have to throw away the idea of "expert systems" intended to replace the experts and get back to "systems for experts", i.e. systems designed to assist experts in a particular field in doing their work efficiently and effectively.

The third priority in the design of Math∫pad was to support the *writing* of scientific articles rather than the production of a finished product. The papers that we ourselves write tend to undergo many revisions, often extending over years rather than months. The drafts we write need to be highly readable but often contain inadequate proofs — inadequate because they are too complicated, not at the right level of abstraction, incomplete, and sometimes downright wrong. When the finished product is finally ready it is usually the case that we feel a sense of shame at our own stupidity during the course of its production. The finished product is intended to show how the ideas should be formulated, and how the theorems should be proved, but it is rarely the case that the finished product represents how its results were first established. The time required for the production of the finished product is also relatively short compared to the time required to reach a state in which a finished product is conceivable.

Verification systems seems to assume that their users are as inhuman as the system itself: always systematic, always tidy and well organised, always precise and always correct. (Or, if this is not the case, at least willing to be disciplined by the machine into such a mode of working.) They are intended to provide the final seal of approval on work that is already well developed and well organised. They do have a purpose but, in our view, a purpose that has been given an unduly high priority. A machine-supplied seal of approval has had no priority in the development of Mathspad.

The final priority in the design of Mathspad was that mathematics and text should be completely integrated. Mathematics is not about mathematical calculations. It is about the nature of the calculations: what the mathematics is about, why one should be interested in a particular calculation or theorem, and what one can learn from it. Doing mathematics is a literary exercise: a mathematical article is an essay, and a mathematics book is a novel complete with a plot and main characters. (We include in this writing computer programs which we also regard as a mathematical exercise.) We cannot envisage doing mathematics without being able to freely intersperse prose with mathematical calculations. Computerised math systems rarely cater for this viewpoint. It is often physically impossible to include explanatory text and where it is possible it is often the case that the inclusion of text is severely restricted.

2.4 Software Tools

One final bit of philosophy: tools are things like hammers and chisels, implements to assist the craftsman. Software tools may be more complicated to produce; nevertheless, their nature is just the same. They are there to assist. They are not there to impose a particular style of working. We have our own views on the scientific method, as will be evident from these pages. You may or may not agree with those views. But, if we have done our job right, it shouldn't matter one way or the other. Mathspad is intended to assist, the way you use it is up to you.

Chapter 3

Tutorial

Like any sophisticated tool, Math∫pad demands an investment of time and effort in order to get used to it. Moreover, because Math∫pad is oriented to the needs of specialists who wish to customise the system to their own specific demands, it can take some time to build up a sufficiently large "project file" to that effect. An analogy is with one's first encounter with the TeX system: so much power at one's finger tips, but at such a low level. The use of LaTeX instead is an improvement, since many more "higher-level" options are built in, but inevitably one has to build up one's own personal macro file before one can effectively and comfortably use the system.

Experienced users of Math∫pad will already have several individually customised "stencils" at their disposal. These files contain specifications of commonly occurring notational elements in the user's documents, ranging from section or chapter headings to the preferred notation and layout for, for example, unbounded summations and products. Before beginning to write a new document the experienced user will —ideally— examine the manuscript carefully for new notational elements and define these appropriately on the system. Few of us, however, are sufficiently well-disciplined to do this in advance, and most will define new stencils on the fly in the process of writing one's document. The greater one's existing stock of definitions the less frequently this is necessary and the smaller the delay it causes. The delay is thus the greatest for the beginner and it is of course the beginner who is the least adept at defining stencils.

In order to ease this initial hurdle a number of stencils are provided with the Math∫pad distribution. The **latex2e** stencil, for example, contains the Math∫pad equivalents of almost all LaTeX2$_\varepsilon$ commands and environments. Since the creation of stencils is the hardest part of using Math∫pad (although it isn't that difficult!) we postpone its discussion to the very end of this tutorial. You will, however, be using various stencils almost from the very beginning even though at times you may not be aware of it.

3.1 The Mouse

Before we can begin at all, it is important that we agree on terminology for referring to the buttons on the mouse. We assume that your mouse has three buttons. If your mouse does not have three buttons you will need to use the rightmost button in combination with the control key to simulate the third button. You are warned that this feature has never been thoroughly tested.

Modern computer systems allow you to customise the mouse (and just about everything else) to your own preferences. For right-handed people the leftmost mouse button is often the one that is easiest to use, for left-handed people (who operate the mouse with their left hand) it is more likely to be the rightmost button. Left-handed people may therefore wish to change the default settings on their computer so that any action initiated by default by the leftmost button is initiated in their case by the rightmost button. We won't tell you how —it's often not very easy to find out and there is no standard (software) mechanism for doing it that we know of— but we are acutely aware that what is ergonomic for one person is not ergonomic for the other.

Ideally, in order to reflect this situation, we would now introduce some (standard) terminology to refer to the three mouse buttons that emphasises their logical functions rather than their relative physical positions. Unfortunately, a standard terminology hasn't yet emerged and until there is we feel it would be too much of a burden to ask you to commit three new terms to memory. What we shall do is very much a compromise: we shall refer to the buttons as the *near* button, the *middle* button, and the *far* button. The near button will typically be —at least for most right-handed people— the leftmost button, and the far button will typically be the rightmost button. (We realise that this terminology may be confusing to those who operate the mouse with their left hand but have not made any changes to the default settings on their computer.)

3.2 The Console

If you are reading this text at your computer, and have so far resisted the temptation to go-it-alone and experiment yourself with the system, then there will be just two windows on your screen, a document window containing this text and the Math∫pad console. The console has seven buttons starting on the left with Window and ending on the right with Quit. Behind each of these buttons is a menu which can be revealed with the aid of the mouse.

Selecting one of these buttons with the far mouse button, with the exception of the Version button, will pull up a menu. Selecting one with the near mouse button will cause a default action to be performed. We do not recommend you do the latter at this stage, especially selecting the Quit button! The Version button may or may not be disabled depending on what you have been doing with this text: more about this later.

All the menus can be pinned up by selecting the menu's title. The complete procedure to pin up a menu is as follows. Move the mouse so that it points at the console button of your choice. Press down the far mouse button — the menu will appear. Keeping the mouse button held down, move the mouse upwards so that the title is highlighted. Finally, release the mouse button. (If you have a colour screen) the menu will get a grey boarder[1] and may then be dragged to any position on the screen you like: depress the near button when the mouse is pointing at one of the boarders and then, keeping the mouse button depressed, move the menu to any position you want on the screen. (If you don't have a colour screen everything will work as just described but there is no visual indication of whether the menu is pinned up or not.)

It is suggested that you pin up three of the menus at this stage: the menus under the EditOp, Structure and Selection buttons. You won't need them all immediately but it may save you time later, and it will give you an idea of the extent of the features offered by Mathſpad. Arrange them as neatly as you can in a clear area of the screen.

(You can get as many copies of the pin-up menus as you wish, and scatter them all over the screen. Mathſpad is not stingy, and tidiness is your concern! To remove a pinned-up menu click on the header with any mouse button.)

In this tutorial we will make use of the Gries-Schneider stencil to explain some of the system's features. This stencil is for illustrative purposes only and you are advised not to use it for any other purpose. You do not need to access the stencil at this point but, if you are curious, you can open it as follows. Pull up the menu under the Window button on the console. Select the Stencil entry by moving the mouse to it and clicking with either the near or far mouse button. A stencil window will appear. Click on the Load button in the top left-hand corner and load the Gries-Schneider stencil by clicking on `Gries-Schneider.mps` in the file selector window in the usual way. (Several more

[1]This varies from window manager to window manager, as does the procedure for moving the menu.

stencils have been used to write this tutorial —indeed, all the stencils listed in the file selector— so do not be mislead into thinking that the Gries-Schneider stencil is all that one needs to prepare a document such as this one.)

Note that we always use Sans Serif characters when we refer to a button on a window or an entry in a menu.

3.3 The Text Editor

Math∫pad recognises the fact that any mathematical document, whether it be a formal proof of some mathematical theorem, the source code of a program, or whatever, must contain provision for the inclusion of plain, non-formal text. To cater for this, there are two distinct modes in Math∫pad: *expression* and *text*. In addition expressions may either be *displayed* or *in-line*. Entering displayed expressions is begun by selecting the Insert Disp item under the EditOp button on the console. Entering in-line expressions is most commonly begun by selecting a template from a stencil but may also be begun by selecting the Insert Expr item. (The latter option is typically used to enter identifiers in the text as in "let x be ...".)

In addition to inserting expressions in text it is also possible to insert text in expressions. As a safeguard, however, it is not possible to do so as a side-effect of an edit operation such as a copy operation. One can explicitly introduce a text place holder in an expression either in a template definition or by choosing the Insert Text item under the EditOp button on the console.

What you are reading now is of course in text mode. Math∫pad incorporates a text editor based on the Emacs editor. There are various commands for moving forward and backwards through the text, for deleting, "yanking" and "killing" text, and so on. If you are familiar with Emacs just try doing what you would normally do and see what happens. Keep your eye on the console for requests for arguments to commands. In addition you may consult chapter 6 for full details. This aspect of the system will not be discussed further here.

3.4 Selections

3.4.1 Clicks and Multiclicks

The main goal of Math∫pad is to enable algebraic calculations to be conducted on a computer terminal. The first thing you must learn in order to do this effectively is how to select subexpressions. A simple mathematical expression has been displayed below in order to let you practise the technique.

$$a + b + c{\cdot}d - (e{-}f){\div}g$$

In Math∫pad there are three types of selection, the target, the source and the argument selection. If there are three buttons on your mouse these will correspond to the near, middle and far mouse buttons, respectively. For the moment it doesn't matter which of the three you use.

The expression above has a certain structure that is governed by the normal precedence conventions of arithmetic. The structure is evident from the spacing around the operators and the parentheses.

Use the scroll bar to position the expression at the very top of the screen so that you can both read and carry out these instructions at the same time. (The mouse pointer should be placed in the scroll bar at the side of the window alongside the expression. Then click with the near button.) Now position the mouse pointer to point to one of the operators and then click (i.e. press and release one of the mouse buttons). Do this for each of the operators in turn. Use the same mouse button each time or else things may get confusing.

In each case the subexpression you have selected will be highlighted (the colour of the highlighting being dependent on the mouse button you used). If, for instance, you click on the division operator you will select the subexpression $(e{-}f){\div}g$. What is selected in this way is the operator and its arguments. Thus, when you click on either of the two "+" operators you select only the two adjacent arguments: clicking on the first "+" selects $a{+}b$ and on the second $b + c{\cdot}d$. Similarly if you click on the leftmost minus sign you select $c{\cdot}d - (e{-}f){\div}g$. If an operator has been declared to be infix (for example "+" or " $-$ ") you can select the whole expression by a "double click" over the operator. Try this with either of the two "+" signs or the leftmost " $-$ " sign. (You must press and release quite fast twice in succession.)

Just for the fun of it, try placing the mouse pointer anywhere in the text and clicking rapidly several times in succession. If you position the pointer in the middle of a word and click twice you will select the word. If you click three times you will select the line in which the word appears, and four times the whole document. You can undo your selection by clicking the same mouse button over a blank line in the text.

The correct technique for multiple clicks so that you can keep track of how many times you have clicked is the following. Position the mouse pointer. Then press a button down but *do not release it*. (That is, keep it held down. Do not allow it to come up again.) When you are ready, release the button and press it in again *rapidly and without releasing*. Pause awhile (keeping the mouse button firmly in place) before repeating the release-and-press-down action. If done at the correct speed you will see the selection widening step-by-step. If you do it too slowly, however, you will go back to square one. At first you will probably be inclined to overdo the speed, or press-down-and-release quickly rather than release-and-press-down quickly, but the technique is soon mastered.

Let us go back to our arithmetic expression. Since it will inevitably have disappeared from your screen by now we reproduce it below so that you can both read and practise simultaneously. Position the text so that the expression is near but not at the very top of the screen. (Leave this sentence visible for example.)

$$a + b + c{\cdot}d - (e{-}f){\div}g$$

Try out your multiple click technique beginning with the mouse pointer positioned over the second minus sign. You will select in turn $e{-}f$, then $(e{-}f){\div}g$ and then the whole expression. It's worth continuing to see what happens next! (Sometimes the selection doesn't seem to alter. If that is the case keep on clicking until the whole screen has been selected.) For most purposes you will only need the single and double click but sometimes more can be useful.

3.4.2 Dragging

Clicking selects complete subtrees of an expression. Dragging the mouse pointer enables you to select certain incomplete subexpressions. Position the mouse pointer this time over b and press a button down again without releasing it. The symbol b will be highlighted. Now *without releasing the button* move the pointer across the screen. Do this slowly (and for as long as you like) so that you can observe its effect. You will find that you can select subexpressions like $b+$, $b + c{\cdot}d\,-$ and $b + c{\cdot}d - (e{-}f){\div}g$. You can also select the subexpression $+b+$ by first positioning the mouse pointer at one of the "$+$" signs and dragging across to the other. You cannot select $b{+}c$ in this way because as soon as you move the pointer over the c the selection jumps so as to include the whole subexpression $c{\cdot}d$. This reflects the way that the expression is stored in the system. It is a tree structure consisting of a root and a *list* of subexpressions. The subexpressions are in this case a, $+$, b, $+$, $c{\cdot}d$, $-$, $(e-f){\div}g$. By dragging you can select any (nonempty) sublist of this list. You can thus also select the individual operators in this way. For instance, positioning the pointer over the symbol "$\div$" and making a very slight dragging motion selects just the operator and nothing more. To select an operator such as the "$\cdot$" which is very narrow and close to its two arguments the best technique is to position the pointer over the operator, and click down but not release. This will select the subexpression $c{\cdot}d$. Then drag *beyond* this subexpression (to include say the adjacent "$+$" in the selection, or even "$b+$") before bringing the pointer back to the "$\cdot$".

Clicking and dragging can be combined, and all that we have said has a similar effect when used to select text rather than subexpressions. Indeed, this is all so much easier to demonstrate than to explain in words, and is not as difficult as it may seem. Selection of subexpressions is a fundamental technique in the use of Math∫pad, so experimentation and perseverance (if necessary) have their rewards. Continue reading only when you feel you have mastered the technique.

3.4.3 Math∫pad Goes Dotty

One of the fine points of selections is to know what exactly it is you have selected! For example, you will have noticed that Math∫pad is capable of handling various fonts: section headings are in **bold** type, names of console buttons are sans serif, and some words are *emphasised*. Now, if you select, say, "*emphasised*" the question is whether you have indeed selected the emphasised word or the unemphasised word "emphasised". Sometimes you will want to select the one, sometimes the other. But which selection has been made is not normally visible.

The problems centre around the use of text as an argument to some function. (Think of emphasising a segment of text. The function is "emphasise" and the segment is the argument.) How can we distinguish between the text itself and the function applied to the text? Sometimes you might not even wish to see a difference in the final text between the argument and the function applied to the argument, for example if you want to index words in a text. During the preparation of the document you will nevertheless most certainly wish to see the difference, in order to be sure that the scope of the function is correctly set. (Those readers familiar with the use of LaTeX will know that a single missing bracket can suddenly turn the whole of their document from roman type into italic type.)

To alleviate these problems Mathpad automatically brackets text arguments by a pair of raised dots. Whether these are visible on your screen can be toggled by holding down the Ctrl key on your keyboard and hitting the "." key. If dots are visible around the several occurrences of Ctrl in this paragraph then holding down the Ctrl key and hitting the "." key will cause them all to disappear. If they are not visible the action will cause them to appear.

Make the dots appear. You can now see whether you have selected the raw word "Ctrl" or the sans serif version by whether your selection excludes or includes the surrounding dots. Moving the mouse pointer to point at the word followed by a double click will give you the raw word, a treble click will give you the sans serif word.

3.4.4 Joining Selections

As mentioned earlier, the three buttons on a mouse are used to make three types of selection. The near mouse button enables you to set the target selection, the middle button the source selection and the far button the argument selection. Joining selections makes use of two types of selection. So now is the time to learn something about the different types. Since the technique we are about to describe is most commonly used when selecting large chunks of text we suggest you practise it on text. Nevertheless, all that we have said so far and are about to say is equally applicable to both text and mathematical expressions.

This is probably an opportune moment to remind you that if you want to clear a selection you should move the mouse pointer to a large area of white space within this edit window and then click on the appropriate mouse button.

Position the screen so that this sentence is at the very top. Now, position the mouse pointer so as to point at the space between "target" and "selection" in the list below. Do a double click with the near button: the effect is to set the target selection to the string "target selection". You can see this because the string is then displayed in inverse video mode. Repeat the action using the middle and far buttons with the pointer positioned between "source" and "selection", and between "argument" and "selection", respectively. On a black-and-white screen, you will see that the source selection is indicated by a thick underline, and the argument selection by a thin underline. On a colour screen the source selection is shaded blue, and the argument selection purple. (These colours may vary, and can be altered to suit your own taste. The important thing is to note for yourself how the different selections are indicated.)

- target selection

- source selection
- argument selection

Selections may overlap in four different ways. See for yourself which colour is used to indicate which combination using the following list as a guide:

- target and source
- target and argument
- source and argument
- target, source and argument

(You can only inspect the combinations one at a time. And, of course, the hard copy of this manual cannot show what appears on the screen if the instructions have been carried out correctly. We suggest you edit the lists above by adding the colour alongside each item. Rename the file to something like "MyTutorial" and save it in your own directory.)

If you now look among all the menus you were asked to pin up earlier you should be able to locate the menu item Join. (If you can't find it, pull up the menu once more by clicking on the Selection button on the console with the far mouse button. Pin the menu up by selecting its title. The selections you have already made will not be lost in this process.) Mostly you will only use the target and source selections. Join is, however, one of the few operations that makes use of the argument selection.

The function of Join is to extend a selection. Next to the word "Join" in the pop-up menu is a triangle pointing to the right. This is the standard convention for indicating that there is a collection of options from which to choose. With the mouse pointing at Join and holding down the far button, drag the mouse to the right and the options will be revealed. Moving the mouse vertically whilst continuing to hold the far button down allows you to choose the option you want.

There are three options to be chosen from. The option "$\mathsf{T{+}A} \rightarrow \mathsf{T}$" joins the target and argument selections. The joined selection becomes the new target. The option "$\mathsf{S{+}A} \rightarrow \mathsf{S}$" joins the source and argument selections, the joined selection becoming the new source. Finally, the option "$\mathsf{A{+}S} \rightarrow \mathsf{A}$" joins the argument and source selections, the joined selection becoming the new argument.

Try it out on the paragraph above. On plain text the effect should be predictable. Note carefully, however, what happens when you join substructures of two disjoint structures. (For instance if you set the target to the second T and the argument to the last S above, and then join them, you will find that the selection immediately extends so as to include the first T and the last A.)

One final point. If you are working simultaneously with two or more documents there is still only one selection of each type available. This is to enable parts of one document to be copied over into, or swapped with, parts of another document. Join does not join selections in different documents! If one of the required selections is not set in the current document then it has no effect.

3.4.5 Clearing and Swapping Selections

There are two other options under the Selection button: Clear and Swap. Both will only work if the menu is pinned up because their action depends on which mouse button is used. Clicking on Clear will clear the selection corresponding to the mouse button that is used. Clicking on Swap with the near button will swap the target and source selections, clicking on Swap with the middle button will swap the source and argument selections, and clicking on Swap with the far button will swap the target and argument selections. It is important to note that swapping *selections* is not the same as swapping the selected *terms*. The latter is executed by using the swap option under the EditOp button and causes the document to change whereas the former has no effect on the document itself.

3.5 Manipulating Subexpressions

The primary goal of Math∫pad is to provide a pleasant, useable interface for performing algebraic calculations at a computer terminal. The argument for using a computer for such calculations rather than doing them by hand is a very simple one: computers are able to copy information very reliably and quickly whereas human beings tend to introduce mistakes in the process.

Apart from being able to copy subexpressions easily and quickly, Math∫pad provides simple-to-use functions for rearranging expressions, and for finding and replacing mathematical expressions of a given structure. There are few restrictions on the use of these functions. The principle is that the user knows what he/she is doing, and Math∫pad is simply there to carry out instructions. There is however one exception, namely, you cannot perform any operation that would result in text being placed where a mathematical expression is expected. For example, there are restrictions on interchanging an expression and a portion of text. This does not mean to say that text cannot appear in mathematical formulae: it is vital that it can. It does mean that, via the stencil definition mechanism, one must explicitly state where text may be inserted in an expression. Bear this in mind when experimenting with the functions described in this section.

The functions provided for expression manipulation are Copy and Swap, Reverse, Distribute and Factorise, Apply, Group and Ungroup, Rename and Find. You will find entries for each of these functions in the various pin-up menus. The functions Swap, Copy and Find can also be used on plain text, but the remainder are peculiar to the manipulation of mathematical expressions. Find is short for Find/Find-and-Replace. We discuss the use of Copy and Swap, Reverse, Distribute and Factorise in this section. Find is discussed in section 3.6.4. Group and Ungroup, Apply and Rename are not discussed in this tutorial. Briefly, Group and Ungroup are used to change the default structure of an expression. Apply is used to "apply" a function to an expression. (The operation is purely syntactic. Don't confuse it with evaluation of the function.) Rename is used to rename dummy variables within a given scope. We refer you to the reference manual for precise details of their use.

3.5.1 Target, Source and Argument Selections

Before proceeding let us recall the three types of selection you can make. These are the target selection, the source selection and the argument selection. These selections are made using the near, middle and far mouse buttons, respectively. Now is the time to appreciate the meaning of these terms.

The target selection is where changes to the text occur. With one exception —the Swap function which swaps the target and source terms— you may rely on the fact that the effect of Math∫pad functions is localised to the target selection only. The source selection determines how the target selection will be affected. The argument selection is typically used to specify the scope of the function that is being applied.

It is important that you accustom yourself to which selection is which. For those readers who skipped over section 3.4.4 here is the information once again: the target selection is indicated on the screen by being displayed in inverse video mode, the source selection is indicated by a thick underscore on a black-and-white monitor and is shaded blue on a colour monitor, and the argument selection by a thin underscore on a black-and-white monitor and purple shading on a colour monitor. Set all three now so that you see what is meant. First set the three selections to *disjoint* areas of text. (Click and drag with the appropriate mouse button anywhere you like.) Then look to see what happens when the selections overlap. (There are four possibilities: overlapping target and source, overlapping target and argument, overlapping source and argument, and all three overlapping. Try out all four.) The effect is self-evident when you see it but much harder to describe in words so we leave it to you to experiment. (That isn't completely true: if you are au fait with systems jargon you will immediately recognise the combination of a number selections as just their exclusive-or.)

3.5.2 Copy and Swap

The edit function that you will use the most is undoubtedly Copy. In this section we describe its operation together with that of the related Swap function. Again, we have an example expression for you to practise on. Position this expression at the top of the screen. (Our apologies to those who are impatient to create their own mathematical expressions. That will be discussed in the next section when we discuss the use of stencils.)

$$a{\cdot}b + c \;\le\; 2{\cdot}\mathbf{E} \;<\; (a{+}b)^3$$

The expression is incomplete: it contains an expression place holder indicated by a boldface **E**. This is just so that you can practise copying. Set the target selection to the "**E**" in the expression and the source selection to any subexpression. Then click on Copy. (If you haven't pinned the appropriate menu up, you will find it under the EditOp button on the console.) The source selection will be copied into the place holder; the target selection becomes the expression you have just copied. If you do not alter the target selection at any stage you can return to the original state by pressing the "Delete" key on your keyboard. Repeat the procedure as often as you like. (Instead of using Copy you may find it more convenient to use the "F7" key on your keyboard. Both have the same effect.) Don't be afraid to copy a subexpression that itself contains the place holder **E**. The copy action is not recursive so you will not send Math∫pad into an infinite loop! Note that Math∫pad sometimes inserts parentheses, and sometimes not. Parentheses are not inserted if you copy an identifier or the subexpression $a{\cdot}b$ into the place holder. The latter is because the multiplication operator has been defined in the stencil for this manual to be an infix operator. Parentheses are inserted if you copy say the subexpression $a{+}b$ into the place holder. This is because "$+$" has been defined to have a lower precedence than "$\cdot$". Note also that the spacing around the operators grows as the expression you create becomes more complex.

You do not need to set the target selection to an expression place holder to perform a copy operation. (What we now have to say holds true in general.) If the target selection is an existing expression then the source selection will simply overwrite the target selection. If, for example, you select $a{\cdot}b$ as the target, and $(a+b)^3$ as the source, copying will result in the expression below.

$$(a+b)^3+c \le 2{\cdot}\mathbf{E} < (a+b)^3$$

The overwritten expression is not lost. It is copied to a buffer. The buffer can be opened by clicking on the Buffer entry (under the Window button on the console). The buffer window is just like a normal document window except that you can't save its contents between sessions. If you have overwritten an expression (or text) by mistake you can always recover it by copying it from the buffer back into your document. (Remember that the different selections do not have to be in the same window.)

As its name suggests, Swap swaps the target and the source terms. (This is the only time that anything other than the target is altered by an edit function.) Swap can be used to swap text or expressions. Text can also be swapped with a mathematical expression provided that the expression is not a subexpression of another expression. (In other words, text and expressions can be swapped only where it is logical to do so.)

Math∫pad doesn't always get the parenthesisation right when new expressions are created. For example, try swapping $a+b$ and the superscript 3 in the expression $(a+b)^3$. Without changing the selections repeat the swap operation. Two swaps like this should have no effect on the text but instead you obtain $a+b^3$. The parentheses around $a+b$ have disappeared! You can't really blame Math∫pad for this because most of us would not wish to parenthesise $a+b$ in the intermediate expression 3^{a+b}. Similarly, if $a+b$ is a subscript to 3 as in ${}_{a+b}3$ we wouldn't want to parenthesise $a+b$. The rules that we use for parenthesisation are non-trivial, and based as much on the visual appearance as the mathematical structure of formulae. Since computers are not endowed with sight one can only expect that things sometimes go wrong. The solution that Math∫pad adopts is to place the responsibility fairly and squarely on the user's shoulders. You are expected to read and check the result of any edit operation you have executed. In the case of missing parentheses there is a menu item Add/Rem () (under the Structure button on the console) that allows you to insert them. Set the target selection to the subexpression you want to parenthesise and then select this item. The parentheses will then appear. If Math∫pad inserts parentheses that you don't want you can remove them with the same menu item. (Sometimes nothing happens when you do this except that the screen flickers. In that case you can oblige Math∫pad to remove the parentheses by holding down the "Control" key on your keyboard whilst clicking on Add/Rem (). The structure of the expression is unchanged by either the insertion or removal of parentheses.) By the way, you can also impose your own will on the spacing within formulae. We won't go into details here —refer to chapter 6 for that— but the general message is: you're the boss and can always override Math∫pad's actions, and you bear the responsibility.

3.5.3 Reverse, Distribute and Factorise

Infix operators play a vital rôle in mathematical formulae. This is due in no small measure to the relative importance of associativity among all the algebraic properties that an operator may enjoy. Because of the associativity of an operator like addition we can freely write, say, $a+b+c+d$ without fear of confusion. This is of great benefit to calculation because the notation is not biased to any particular grouping of the subexpressions. The exploitation of associativity in a calculation becomes an invisible step, one that we are often not consciously aware of.

A similar phenomenon lies at the heart of the use of infix notation for transitive relations. When we write, say, $a = b = c$ we mean $a = b$ *and* $b = c$. But, because of the transitivity of equality, it is also the case that $a = c$. In an extended calculation in which several steps occur it is typically the equality between the first and last terms that one is interested in, the intermediate terms being only of passing interest. At the end of such a calculation it is nonetheless rare to see explicitly stated "and hence, *by transitivity of equality,* the first and last terms are equal". Such would only happen in a very elementary mathematical text, or in an extremely detailed (and pedantic) logical argument. The infix notation encourages us to *associate* the first and last terms with each other.

Several elements of Math∫pad are geared to the use of infix notation in the two ways explained above. When creating a stencil it is possible to declare that an operator is "infix". This means that the operator can have a list of arguments, of arbitrary finite length, whereby the notation used to denote the application of the operator to a list is to repeatedly write the operator between successive elements in the list of arguments. Thus $a + b + c$ denotes the application of the operator $+$ to the list $\{a\ ,b\ ,c\}$. Equally, $a = b = c$ denotes the application of the operator $=$ to the list $\{a\ ,b\ ,c\}$. These are both called "associative infix operators" in Math∫pad jargon because the notation is intended to encourage one to "associate" terms in the expression together in a variety of different ways.

(It is also possible to declare an operator to be "left associative" or "right associative". Such a declaration has the conventional meaning. For example, if the operator $\Rightarrow$ has been declared to be right associative then the expression $a \Rightarrow b \Rightarrow c$ has the structure $a \Rightarrow (b \Rightarrow c)$. If you click on the leftmost "$\Rightarrow$" you will select the whole expression, and you cannot select the subexpression $a \Rightarrow b$. You can, however, select $a \Rightarrow b$ by dragging. This should be contrasted with the situation in which $\Rightarrow$ is declared to be associative. Clicking on the leftmost "$\Rightarrow$" selects $a \Rightarrow b$ whilst dragging or a double click is needed to select the whole expression. This is the only difference between "left/right associative" operators and "infix" operators in Math∫pad. Our own advice would be to declare all binary operators to be plain "infix" rather than specifically left or right associative, but the latter feature has been retained for those who do not wish to follow our advice.)

The functions that exploit the notion of associativity are Reverse (called Commute in older versions of the system), Distribute and its inverse Factorise. (These functions are all to be found in the pin-up menu under the Structure button on the console.)

The function Reverse reverses the order of the elements in a list of arguments and (associative) infix operators. Try it out on the following expression:

$$a \leq b < c = d \leq e$$

First, by pointing at the individual operators in turn and clicking, note that there is no hidden tree structure present in the expression. It is just a list of variables (a to e) separated by comparison operators. By dragging you may therefore select any sublist of the list. For the moment don't try to be nasty; select a sublist that begins and ends with a variable. Then click on Reverse. All the elements of the list you have selected will be reversed *including the comparison operators*. Thus, if you select the subexpression $a \leq b < c$ the effect will be to create the following expression.

$$c < b \leq a = d \leq e$$

Note that not only has the order of a, b and c been reversed but also the order of $\leq$ and $<$. Now do try to be nasty. Select a sublist that doesn't begin and end with a variable. You will discover that the syntactic structure —expressions separated by operators— remains intact. Within this constraint Math∫pad reverses as much as possible of the list you have selected.

You may be disappointed that Math∫pad does not also reverse the symbols used to denote the comparison operators as well. (Thus reversing the whole expression should give $e \geq d = c > b \geq a$.) Such a feature has not been implemented. You can always edit the individual operators yourself by hand after having reversed the list. It is sometimes tedious, but it can be useful to check through what you have got as you do it.

Here is another expression for you to experiment on. The point to note is that Reverse is not recursive: it does not reverse subexpressions of the expression you have selected.

$$a{\cdot}b + (c{\div}d){\cdot}e + f$$

So much for the Reverse function.Let us now discuss Distribute and Factorise. These two are most often used in the normal situation in a calculation where a function is known to distribute over a binary operator, for instance to take the step from $a{\cdot}(b{+}c{+}d)$ to $a{\cdot}b + a{\cdot}c + a{\cdot}d$ ("distribute") or vice-versa to go from $a{\cdot}b + a{\cdot}c + a{\cdot}d$ to $a{\cdot}(b{+}c{+}d)$ ("factorise"). The functions can be useful in other situations too. One such situation arises when attempting to prove an equality, say $a = c$. Not knowing from which side to begin, or suspecting that induction may be necessary at some stage, one might hedge one's bets and calculate with the equality $a = c$ as follows.

$$\begin{array}{ll}
 & a = c \\
\equiv & \qquad \{ \qquad \text{reason why } a = b \qquad \} \\
 & b = c \\
\equiv & \qquad \{ \qquad \text{reason why } b = c \qquad \} \\
 & c = c
\end{array}$$

It is no hardship when working with Math∫pad to repeatedly copy "$=c$" even if the expression "c" is quite complicated. Doing so also helps you to keep track of what it is you are heading for. On completion of the calculation, however, you may decide that it would be more helpful to a reader if the "$=c$" were omitted. You can do this as follows. First, factorise "$=c$" from the expression (using the technique to be explained shortly). This results in

$$\begin{array}{ll}
 & (a \\
\equiv & \qquad \{ \qquad \text{reason why } a = b \qquad \} \\
 & b \\
\equiv & \qquad \{ \qquad \text{reason why } b = c \qquad \} \\
 & c) = c
\end{array}$$

Then set the target selection to the whole expression and the source selection to the subexpression within parentheses and click on Copy. You obtain:

$$
\begin{array}{lll}
 & a & \\
\equiv & & \{ \quad \text{reason why } a = b \quad \} \\
 & b & \\
\equiv & & \{ \quad \text{reason why } b = c \quad \} \\
 & c &
\end{array}
$$

Now with Find (to be explained in section 3.6.4) you can replace all occurrences of "$\equiv$" in your proof by "$=$" and you are done.

The fact that the intermediate steps in this edit process do not make mathematical sense should be of no concern to you. Indeed in order to exploit the capabilities of Math∫pad to the full you must learn to view calculation purely and simply as a game with (structured) sets of meaningless symbols.

The simplest way to use Distribute is as follows. Suppose you have an expression consisting of a function applied to an expression where the argument to the function is some infix operator applied to a list of expressions. For example, the expression

$$a\cdot(b+c+d)$$

comprises the function $a\cdot$ applied to the argument $b+c+d$. Set the target selection to the whole expression and the source selection to the function's argument. In the above example set the target to $a\cdot(b+c+d)$ and the source to $b+c+d$. Ensure that the argument *selection* is not set. (Otherwise the procedure described in the next paragraph will be followed.) Now click on Distribute in the console. The target will be modified in such a way that the function is "distributed" over the argument. Try it out on the expression displayed above. You should get what you expect to get, namely $a\cdot b + a\cdot c + a\cdot d$.

The simple procedure just described is not adequate in all circumstances. In general, there are two components to a Distribute or Factorise operation, the target selection and a *function* that is distributed over, or factorised out of, the target selection. Sometimes, as in the example above, the function is a part of the target itself, but this may not always be the case. In the more general case both the source and argument selections are used to specify the function. The two are set so that the argument selection is a subexpression of the source selection; the function is then the difference between the two. Try this out as follows. Set the target selection to the entire expression displayed below. Set the source selection to $a\cdot b$ and the argument selection to b (thus setting the function to $a\cdot$). Now click on Factorise.

$$a{\cdot}b + a{\cdot}c + a{\cdot}d$$

If you have done it correctly the expression above will now be identical to the expression below.

$$a{\cdot}(b{+}c{+}d)$$

Try reversing the process as follows. Reset the target and argument selections to the expression $b{+}c{+}d$ (double click somewhere within the expression) and set the source to the whole expression. Now click on Distribute.

Surprise, surprise! If you thought you would get $a{\cdot}b + a{\cdot}c + a{\cdot}d$ you are wrong! What in fact happens is that you obtain $a{\cdot}(a{\cdot}b + a{\cdot}c + a{\cdot}d)$. This is not a bug. The principle is that only the *target* selection is affected by any Math∫pad function. The target selection was $b{+}c{+}d$ and this has been replaced by $a{\cdot}b + a{\cdot}c + a{\cdot}d$ as requested. The initial $a{\cdot}$, selected as part of the source selection in order to define the function to be distributed, has been left unchanged in the process. You can easily return the document to its original state by now setting the target to $a{\cdot}(a{\cdot}b + a{\cdot}c + a{\cdot}d)$, the source to $a{\cdot}b + a{\cdot}c + a{\cdot}d$ and clicking on Copy.

The general rule is that the function part of a Factorise or Distribute operation is the difference between the source and the argument selections when the latter is a subexpression of the former. If the argument selection is not a subexpression of the source (including when the argument is not set) then the source must be a subexpression of the target, and the difference between these two is used as the function. The source must always be a complete expression. Thus in the above example it is not possible to define the function part by setting the source to $a{\cdot}$.

This method of specifying a function is typically very convenient but it is of course not complete. For instance it is not possible to specify the function mapping x to $x{+}x$ in this way. In such circumstances you will have to perform the operation by hand. Sorry!

There is another aspect of Factorise that may at first seem surprising. Suppose we take as target the expression

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

With the aid of the source and argument selections, set "$a{\cdot}$" as the function to be factorised out. Now click on Factorise. One might expect Math∫pad to refuse in some way — an angry beep or rebuff of the sort "How stupid can you get, I can't factorise the given subexpression" (but perhaps more tactfully stated). Instead one obtains the expression:

$$a \cdot (b+c)$$

Try it again with the following target expressions

$$a \cdot b + c + a \cdot d + e$$

$$b+c$$

In each case an expression is created of the form $a \cdot \mathbf{E}$ where the subexpression $\mathbf{E}$ is the original target with all occurrences of "$a \cdot$" removed.

The design decision underlying this effect is that Math∫pad functions should be *total*, so far as possible. No angry beeps, no remonstrances. We, as designers of Math∫pad, cannot predict how others might wish to exploit the system, and we certainly do not wish to *dictate* how the system should be used.

We conclude this discussion of distribution and factorisation with a slightly more complicated example. Suppose we want to include a step in a calculation of the following form:

$$\begin{array}{ll} & \{a\ ,\ b\} \\ = & \qquad \{ \qquad \text{set calculus} \qquad \} \\ & \{a\} \cup \{b\} \end{array}$$

We assume that you have a set-calculus stencil for all the notational elements in the calculation. Using your stencil you can, of course, enter the whole step by hand, not making any use of Distribute. There may however be circumstances in which that would be quite cumbersome to do — for example if the list of elements in the set is quite long and contains complicated expressions.

The step clearly involves a distributivity property, but not so straightforward as distributivity of multiplication over addition. The way to proceed is as follows. In your calculation you will already have the line

$$\{a\,,\,b\}$$

You would now extend the calculation by choosing in a "proof" stencil your own favourite notation for a calculational step. The notation used above is the favourite of the designers of Math∫pad so that is what we will use here. Taking the appropriate action (i.e. selecting the expression with the target selection and clicking on the correct notation) we get the template

$$\begin{array}{ll} & \{a\,,\,b\} \\ \mathbf{O} & \qquad \{\quad \mathbf{T} \quad\} \\ & \mathbf{E} \end{array}$$

in which all of **O**, **T** and **E** are place holders (for operator, text and expression, respectively). The equality sign, the hint and the subexpression " $a\,,\,b$ " are now filled in so that the state of the calculation is as follows:

$$\begin{array}{ll} & \{a\,,\,b\} \\ = & \qquad \{\quad \text{set calculus} \quad\} \\ & a\,,\,b \end{array}$$

Now we apply Distribute. The target is set to " $a\,,\,b$ " in the *bottom* line of the calculation, the source to " $\{a\,,\,b\}$ " and the argument to " $a\,,\,b$ " in the *top* line of the calculation, and Distribute is activated. (Try it.) The result is:

$$\begin{array}{ll} & \{a\,,\,b\} \\ = & \qquad \{\quad \text{set calculus} \quad\} \\ & \{a\}\,,\,\{b\} \end{array}$$

Not quite what we wanted. But now the comma in the bottom line can be selected (point at it and then drag a slight distance) and deleted (press the "Backspace" or "Delete" key on your keyboard). The operator place holder that then appears can be filled in using a set-calculus stencil to obtain the desired result, namely:

$$\begin{array}{ll} & \{a\,,\,b\} \\ = & \qquad \{\quad \text{set calculus} \quad\} \\ & \{a\} \cup \{b\} \end{array}$$

3.6 Stencils

We now come to the most important element of Math∫pad, the so-called stencils.

A stencil is a file describing the notational conventions used in one or more documents, and how these are to be set using the LaTeX document preparation system (or other ascii-based mark-up language). Each stencil consists of a number of templates, each of which may have various versions.

In this section we discuss in some detail how to *use* existing stencils, and in much less detail how to *create* new stencils.

3.6.1 Using Stencils

In order to illustrate the use of stencils we have extracted various notational elements from the book "A Logical Approach to Discrete Math" by David Gries and Fred B. Schneider and incorporated them in the Gries-Schneider stencil. If the stencil is not already visible on your screen you should call it up now. To do so pull up the menu under the Window button in the console and click on Stencil. Click on Load in the stencil window and then choose `Gries-Schneider.mps` from the list of files in the file selector window.

Like all authors, Gries and Schneider have their own style for presenting definitions, theorems etc., and their own notational conventions for constructing mathematical expressions. The Gries-Schneider stencil is in effect a syntax-directed editor for statements in their book. (This is, of course, only partially true. The stencil we have created contains only a small number of templates in order to illustrate the general idea. Many more would be needed for the complete book. More importantly, not all the conventions used by Gries and Schneider are peculiar to their own book. For organisational reasons it is better to create several stencils, each forming a logical unit. For the examples below we have made use of other stencils, in particular the latex stencil, in addition to the Gries-Schneider stencil.)

Figure 3.1 is an extract from page 151 of their book where the rule "change of dummy" is stated and proved. The extract is not completely identical to what appears in their book, but almost.[2] We will explain the differences at the appropriate time.

Since the extract is rather long it will be impossible for you to keep it in view as you

[2]We assume you are reading this document on screen. If you are reading the hard copy version it may be difficult to appreciate some of the remarks made later.

(8.22) **Name** **Change of dummy**:
Condition: Provided $\neg occurs(`y' \ , \ `R\,, P')$ and f has an inverse,
Statement: $(\star x \ \mid \ R \ : \ P) = (\star y \ \mid \ R[x := f.y] \ : \ P[x := f.y])$

The proof of this theorem illustrates the use of several of the axioms given above. The proof starts with the RHS of (8.22), because it has more structure.

$$
\begin{array}{ll}
 & (\star y \ \mid \ R[x := f.y] \ : \ P[x := f.y]) \\
= & \langle \quad \text{One point rule (8.14) — Quantification over } x \text{ has to be introduced. The One-point rule is the } \textit{only} \text{ rule that can be applied at first.} \quad \rangle \\
 & (\star y \ \mid \ R[x := f.y] \ : \ (\star x \ \mid \ x = f.y \ : \ P)) \\
= & \langle \quad \text{Nesting (8.20) — Moving dummy } x \text{ to the outside gets us closer to the final form.} \quad \rangle \\
 & (\star x, y \ \mid \ R[x := f.y] \wedge x = f.y \ : \ P) \\
\mathbf{O} & \langle \quad \mathbf{T} \quad \rangle \\
 & \mathbf{E}
\end{array}
$$

Figure 3.1: An Extract from Gries and Schneider's Book

read. A copy has therefore been placed in a separate file called "GriesSchneiderExample". We suggest you pull up this file and place it alongside the current one. In case you have forgotten here once more are the instructions for doing so: Select Document in the menu under Window on the console and, when a window appears, click on the Load button. Finally, select the entry `GriesSchneiderExample.mpd` from the list of files that appears.

If you look at the Gries-Schneider stencil you will see that it has six sections, labelled <u>None</u>, <u>Prefix</u>, <u>Postfix</u>, <u>Infix</u>, <u>Left</u> and <u>Right</u>. All but the first of these sections define mathematical operators and their precedences. The first section (the one labelled <u>None</u>) is used to define mathematical notations *and* other (not necessarily mathematical) conventions that are used in a document.

Ostensibly, nine conventional operators have been defined in the stencil — reading from top to bottom, $\neg$, $\equiv$, $\wedge$, $=$, $+$, mul , . , $\Leftarrow$ and $\Rightarrow$. (The unconventional operators $[\mathbf{V} := \mathbf{E}]$ and $\mathbf{O}\,\langle\,\mathbf{T}\,\rangle$ will be discussed shortly.) The first is a prefix operator, the next five are infix operators, the next two are left-associative infix operators, and the last a right-associative operator. We say "ostensibly" because several of the operators have different "versions". If for instance you move the mouse to point at " $\wedge$ " in the stencil and press down with the *far* mouse button a small menu will appear containing two items " $\wedge$ " and " $\vee$ ". The two operators have different meanings, of course, but according to the definition in the stencil, they are treated by Math∫pad as different "versions" of the same operator. In fact the total number of operators defined in the stencil is seventeen. (Go through each in turn to see which have and which do not have different versions. Before you can proceed you must either pin the menu up —using the by now familiar technique of selecting the title— or dismiss the menu by pointing at one of the entries, pressing down the far mouse button, and then moving the mouse pointer outside the menu before releasing the mouse button.)

Each operator has been assigned a precedence according to a list to be found at the beginning of Gries and Schneider's book[3]. Different versions of an operator all have the same precedence. Although we don't discuss how here, it is very easy to add new operators to the stencil or to modify existing definitions. Doing so results in a simple syntax-directed editor for mathematical expressions. For instance, the expression $\neg(p{\Leftarrow}q \wedge r)$ can be created either top-down or linearly as follows:

Top down: (Using the near mouse button) click on " $\neg$ " in the stencil. This creates the expression $\neg\mathbf{E}$ at the position of the cursor in your document. The target selection will automatically be set to the place holder "**E**". Now click on " $\wedge$ " in the stencil. Parentheses are automatically inserted (because the precedence of " $\wedge$ " has been defined to be less than that of " $\neg$ ") and the expression will have been transformed to $\neg(\mathbf{E}{\wedge}\mathbf{E})$. The target selection will be automatically set to the first expression place holder. Click on " $\Leftarrow$ " in the stencil. The expression is transformed to $\neg(\mathbf{E}{\Leftarrow}\mathbf{E} \wedge \mathbf{E})$. No parentheses are introduced because " $\Leftarrow$ " has been defined to have a higher precedence than " $\wedge$ ". (Not a choice we would make, by the way, but everyone to their own taste!) The spacing around the " $\wedge$ " sign, however, increases automatically. Fill in "p" and press the space bar or "Enter". The target selection automatically switches to the second place holder. Fill in "q" and press the space bar or "Enter". Finally fill in "r". The construction is complete when you press the space bar or "Enter".

Left-to-right: Begin as before by clicking on " $\neg$ " in the stencil. Now key in an opening parenthesis. You will find that the closing parenthesis is automatically inserted giving you $\neg(\mathbf{E})$. The target selection is also automatically set to the expression holder. Key in "p" but this time do not press the space bar (or "Enter"). Click on " $\Leftarrow$ " in the stencil and then key in "q". Now click on " $\wedge$ " in the stencil. Key in "r". Again, the

[3]The entry "mul" deviates slightly from their book since it gives multiplication and division the same precedence. This is of no relevance here.

construction is completed when you press the space bar or "Enter" on your keyboard. (If, instead of pressing the space bar or "Enter", you key in a closing parenthesis the target selection will jump to the subexpression $(p \Leftarrow q) \wedge r$ in anticipation of your continuing this particular subexpression. This is not intended — it should jump to the subexpression $\neg((p \Leftarrow q) \wedge r)$. To continue the latter you have to reset the target selection yourself by hand.)

The first entry in the None section of the Gries-Schneider stencil is labelled "rule". This template describes the style used by Gries and Schneider to present rules. Each rule typically has a *label*, a *name*, a *condition* or context under which the rule is valid, and the *statement* of the rule itself. In the above example the label is "8.22", the name is "**Change of dummy**", the condition is "Provided $\neg occurs(\text{'}y\text{'}\ ,\ \text{'}R\ , P\text{'})$ and f has an inverse", and the statement is "$(\star x \ \mid\ R\ :\ P) = (\star y \ \mid\ R[x := f.y]\ :\ P[x := f.y])$".

Clicking on "rule" in the stencil with the near mouse button causes the following template to appear in your document at the location of the target selection:

(**label**)	**Name**	**T**:
Condition:		**T**,
Statement:		**E**

The bold "**label**" and the two bold "**T**" 's are place holders for text; the bold "**E**" is a place holder for a mathematical expression.

This template illustrates two ways of reminding oneself of the function of a place holder in a template. Both ways are intended to facilitate the process of *writing* a document without compromising the quality of the finished product.

The first way is to give place holders a name: "**label**" is a named place holder and can be replaced by an arbitrary text. It reminds the writer that a label is needed and where it should be entered.

A characteristic of named place holders is that the name disappears as soon as anything is entered into the place holder. This can be an advantage, for instance in order to avoid clutter in the on-screen view of a document, but it can also be a disadvantage. In the latter case one can add explanatory information to the screen view of a template which is then discarded in the generated output. This, the second way of reminding oneself of the function of a place holder, is illustrated by the use of the headings "**Name**", "Condition:" and "Statement:" together with the unnamed place holders "**T**" and "**E**". In Gries and Schneider's book these headings do not appear. The point is that the template can always be defined so that the headings *do* appear on the screen but do *not* appear in the LaTeX output generated by the document.

It is sometimes the case that Gries and Schneider do not give a rule a name, or the rule is unconditionally valid. To accommodate these additional possibilities the template has three versions. You can see these versions by pointing at "rule" in the stencil, and then clicking with the far mouse button. A menu will appear with three entries Name, Condition, Statement, Condition, Statement and Name, Statement. (There is also an entry Info separated from the rest which, if selected, provides further information on the template. We return to this later.) We suggest that you pin this menu up. If you now press down on either the near or far mouse button you can move the pointer to select one of the entries. If you release the mouse button whilst one of the items is selected then a template will appear in your document at the position of the cursor. (The menu will remain on your screen if you had pinned it up earlier and will disappear if you hadn't pinned it up.) If you release when none of the items is selected then the menu will disappear and nothing will happen to your document.

The second entry in the Gries-Schneider stencil is labelled "Quant.". This too has several versions which you can examine by pointing at the entry and pressing down on the far mouse button. In the example above we in fact only use one of the versions, namely (**EV** | **E** : **E**). This version has place holders for four entries, an **E**xpression, a **V**ariable, and two **E**xpressions. Try selecting different components of the template displayed below:

$$(\mathbf{EV} \mid \mathbf{E} : \mathbf{E})$$

You will find that you are unable to select the "syntactic sugar": the individual parentheses, the vertical bar and the colon. Only the four place holders and the entire expression can be selected.

Now set one of the expression place holders to be the target selection. Then, with the near mouse button, click on any entry in the stencil. Repeat this as often as you like and observe how the expression grows. Observe also which parts of the expression can be selected and which not. (If you are unfamiliar with the term, this is what is meant by "syntax-directed editing".)

The next entry in the <u>None</u> section of the stencil is labelled $\mathbf{E}(\mathbf{E})$. Clicking on it (in the stencil window) with the near mouse button generates the template $\mathbf{E}(\mathbf{E})$ at the position of the cursor in your document. This template has been used to construct the expression $occurs(\text{'}y\text{'}\ ,\ \text{'}R\ ,\ P\text{'})$. Specifically, the first place holder has been replaced by the expression $occurs$ and the second place holder by the expression $\text{'}y\text{'}\ ,\ \text{'}R\ ,\ P\text{'}$.

It is useful to compare the template $\mathbf{E}(\mathbf{E})$ with the template labelled $[\mathbf{V} := \mathbf{E}]$ in the stencil. Clicking on the latter generates $\mathbf{E}[\mathbf{V} := \mathbf{E}]$ at the position of the cursor in your document. The additional " $\mathbf{E}$ " at the beginning is implicit in the fact that $[\mathbf{V} := \mathbf{E}]$ is a postfix operator. (Note: Operators in Math∫pad needn't be single symbols; as illustrated here, they may be expressions with arbitrary structure.)

The postfix operator $[\mathbf{V} := \mathbf{E}]$ has been defined to have the highest precedence of all the operators in the Gries-Schneider stencil. (You can't see this from the information displayed on your screen. You have to look at the "Info" to find that out.) This means that whatever expression is entered as argument to the operator it will be automatically parenthesised. (There are two exceptions: if the expression is just an identifier and if the expression is an instance of the same postfix operator.) Expressions that are entered in the first place holder of the template labelled $\mathbf{E}(\mathbf{E})$ will never be parenthesised by Math∫pad. The point is that the automatic insertion of parentheses is applicable only to the templates defined outwith the None section. Math∫pad will never try to parenthesise an expression that occupies a place holder of a template in the None section. Typically, mathematical operators defined in the None section are multi-fix operators, and it is advisable as a general rule to ensure that every place holder is bracketed on both sides by some "syntactic sugar". (This is exemplified by the quantifier template.) If this is unavoidable an effect will be obtained that is comparable to defining a postfix, prefix or infix operator with the lowest possible precedence. If in such cases you feel it would be more helpful to your reader to insert parentheses this can always be done using the Add/Rem () item in the menu under Structure on the console.

Instead of defining a postfix operator $[\mathbf{V} := \mathbf{E}]$ we could have defined a postfix operator $[\mathbf{E}]$ combined with an infix operator $:=$. This is more flexible but has the disadvantage that more hand movements are necessary to enter an expression. A better solution is to enter both as "versions" of a template as illustrated by the quantifier template discussed above. This is one argument for the use of versions. Frequently used instances of a general construction can be entered as versions of the construction and more quickly entered into a document than by entering the general version followed by the details of the instantiation. The main argument for versions however is organisational. It is remarkable just how many different templates one needs in even a small document and the total size of one's stencil database grows very rapidly. Apart from the fact that a computer screen is much, much smaller than a desk and can very quickly become clogged up, looking up the template you want can become time-consuming. Sensible use of versions and logically-organised stencils can make a world of difference. Math∫pad provides the tools. How well (or badly) they are used is up to you.

The final operator in the Gries-Schneider stencil has the label "$\mathbf{O}\,\langle\,\mathbf{T}\,\rangle$". This is an excellent example of the flexibility of the template notion in Math∫pad.

The template cannot be used for in-line mathematical expressions; it can only be used in display mode. This is because its definition includes specific layout requirements, in particular that the operator's two arguments are to be placed one above the other on separate lines.

To create a displayed expression you should set the target selection to an empty line where you want the expression to be displayed. Then select the entry Insert Disp under the EditOp button on the console. A bold-faced **D** will appear in the text like the one below. The target selection will be set to the "**D**".

$$\mathbf{D}$$

You don't need to create a new display in this document. Just set the target selection to the displayed "**D**" above. Now click on $\mathbf{O}\langle\,\mathbf{T}\,\rangle$ in the Gries-Schneider stencil. This is what you should get.

$$\begin{array}{lcl} & \mathbf{E} & \\ \mathbf{O} & & \langle \quad \mathbf{T} \quad \rangle \\ & \mathbf{E} & \end{array}$$

A place holder has been created for two expressions (**E**), an operator (**O**) and text (**T**). If you continue clicking on the same template you will find that the pattern is repeated. Each time, you get one each of an extra expression, operator and text place holder. The template $\mathbf{O}\langle\,\mathbf{T}\,\rangle$ behaves exactly like an infix operator. Its precedence has however been defined to be 0, the lowest possible value. If you enter a particular operator in the operator place holder (just click on one in the stencil) then the precedence remains 0; the construction does not adopt the precedence of the instantiated operator.

3.6.2 Help with Templates

It is often the case that you want help with the use of templates, even with templates that you yourself have created. The most comprehensive information is provided in the definition of a template. See the next subsection for how to redefine existing templates. By going through the motions of redefining a template, but not making any change whatsoever, you can find out full details of any template.

Two other mechanisms are provided in order to let you obtain information about existing templates. The first is obtained via a selection in a Document window, the second via a stencil. We consider both in turn.

Suppose you are reading a Math∫pad document, possibly written by someone else, and you see that use has been made of a template. You would like to find out more about the template, for instance, in which stencil (or stencils) it is defined. In this document, for example, you will no doubt have noted that a template has been used for sections and subsections.

To find out more, set the target selection to the template you are interested in. (You may sometimes have to move up or down in the tree structure to find the correct template.) Then select Version in the console with the far mouse button. Pin up the menu that appears and select Info. The information that appears contains the following items:

- The name of the template.
- The name of a help file.
- The type (mathematical operator or not) and, if a mathematical operator, its precedence and default spacing.
- The number of versions of the template.
- The number of times the template has been used.
- The names of loaded stencils in which the template can be found.

The name of the template is the entry name under which the template appears in any stencil.

A help file associated with a template may or may not have been provided by the creator of that template. The contents of a help file is also the responsibility of the creator of the template. The intention, however, is that the help file explains the purpose of the template and gives examples of its use. A help file is just a normal Math∫pad document and so can be read by loading it via the Document entry below the Window button on the console.

The template type is either “none”, “prefix”, “postfix”, “infix”, “left” (meaning “left associative”) or “right” (meaning “right associative”). The precedence and spacing are not relevant in the first case. In the other cases the precedence is a number between 0 and 20 expressing the relative binding power of the operator. (We recommend that all prefix and postfix operators are given the highest precedence. The precedence of infix operators is of your own choosing, but a good rule is that the smaller the symbol used to denote the operator the higher its precedence should be. Thus an infix dot should have a high precedence whereas a long arrow should have a low precedence.)

The number of versions of the template is self-explanatory. The number of times the template has been used relates to the documents that are loaded at the time the information is requested. It also includes uses of the template in the Buffer window and in the Find window. (Thus, if you delete a template from a document the number of uses will not change because the template is automatically deposited in the buffer.)

Templates may be stored in more than one stencil. The section "Files" lists the names of all the stencils that are currently loaded in which the template has been defined. Often a dash (–) will appear in this entry indicating that the template doesn't occur in any loaded stencil. This doesn't mean that the template is nowhere defined, only that you haven't loaded any stencil containing its definition. If the creator of the template has been conscientious then the name of the stencil will be included in the help file. So, consult the help file first if the template does not occur in any loaded stencil. Otherwise the definition of the template is always accessible via the so-called "document stencil": see section 8.8.1 for further details.

If you have the appropriate stencil loaded then the same information can be obtained by selecting a template with the far mouse button and then selecting Info in the menu that appears.

Note that information is available on all templates, even if the template has only one version.

3.6.3 Creating Stencils

The creation of (versions of) a template in a stencil can be very straightforward; it can also be the most difficult task of all in the use of Math∫pad. It is very straightforward to enter simple mathematical operators. The difficulties begin when you require special layout conventions. The difficulties are compounded when the LaTeX automatically generated by Math∫pad is not adequate to your needs. In this section we only outline the creation of templates and you are referred to the reference section for more details.

To create a new stencil click on Stencil in the menu under Window on the console. Once the stencil window appears you can change its default name to whatever you like using Rename.

To create a new template or amend an existing template click on Define at the head of the stencil window. At the same time you will probably wish to open the symbol palette by clicking on Symbol in the menu under Window on the console. This window contains several pages of mathematical symbols which you can enter directly into your documents or use in the definition of stencils.

Defining templates is menu-driven. The window opened by clicking on Define is headed by a list of items. The first three ("Kind", "Prec." and "Space") must always be set. When you open the window these items are already initialised to the default values "None", 0, and 0.

To enter, for example, the operators $\otimes$, $\oplus$, and $\oslash$ all as versions of one template called, say, "o.ops" proceed further as follows. First enter the name "o.ops". (You can skip this step if you want. If you don't supply a name Math∫pad will create one for you. All the names in the Gries-Schneider stencil were created by Math∫pad except for "rule", "Quant." and "mul".)

Now select "Kind" and choose an item (for example "Infix") from the menu that appears. After having done so select "Prec." and choose a number from the menu that appears.

Now move the mouse pointer down into the main window area and click. The three headings "Screen", "Latex" and "Name" will appear. If you move the mouse pointer below all three and click again a second such entry will appear. Each of these entries forms a version of the template you are defining.

The "Name" field allows you to give an individual name to each version. (In this case it will not be necessary to do that since the name Math∫pad creates is perfectly adequate.) The "Screen" field lets you specify how the (version of the) template is to be laid out on the screen, and the "Latex" field specifies the LaTeX code that is generated by each version of the template in your document.

For our example you would click in the "Screen" field in order to indicate that it is that field you wish to edit. Two cursors appear, one in the "Screen" field, the other in the "Latex" field. The two cursors indicate that Math∫pad will automatically generate LaTeX corresponding to whatever is entered in the screen filed. Then you would look up the symbol $\otimes$ in the symbol palette and click on it. The symbol is copied to the "Screen" entry and the corresponding LaTeX is automatically filled in in the "Latex" field. This process is then repeated for $\oplus$ and $\oslash$.

Clicking on Install followed by Done in the Define window completes the operation.

The limitations of the written word have obliged us to describe the above process as a *sequence* of steps. The steps may however be performed in any order.

The remaining menu entries allow you to specify the default spacing around operators, to align different elements of a template, to specify the kind, size and/or font of individual place holders, and to draw variable length lines alongside, above or below the elements of the template. It is also possible to hide elements of a template on the screen (but then you should include a version of the template in which the element is not hidden so that you can edit it!).

In more complicated cases involving, for example, alignment of different elements or sub- or superscripting, the best advice is to look for a template that approximates the one that you want, and use that template as a blue-print. You can include an existing template in a new one just by clicking on the template when the cursor is set in the Define window. The complete definition will be copied across.

The three other buttons at the head of the Define window — Modify, Delete and Double allow you to edit a stencil. Delete deletes the template that is loaded in the Define window, and Modify changes the definition of the loaded template. (You can see which template is loaded by looking at the "Name" entry.) Take care with Modify and Delete since there is no undo facility!

The Double button allows you to remove a duplicated entry in a stencil whilst at the same time ensuring that all occurrences of the duplicate in loaded documents are identified with the original. This is useful in case you need to integrate two or more independently produced documents. Suppose, for instance, you are writing a document in collaboration with a colleague. Both you and your colleague define independently a "woggle" template, say, for use in the sections of the document on which you are working. Subsequently, the two sections are to be merged. The problem that you then face is that, although the two woggle templates may produce exactly the same output, Math∫pad does not recognise them as being the same. (Math∫pad uses a code number to identify templates, the number being generated at the time of creation of the template. The chances that the code number given to two templates is the same is negligibly small.) Any find operation, for example, will find only one of them. The solution is to first ensure that the two templates are indeed identically defined and then, with one of the templates loaded in the Define window, click on Double. This will update the code number assigned to the loaded template to be identical to the code number assigned to the other, both in the stencil and in all loaded documents. Deleting the template from the stencil leaves you with just the one instance, as required. Refer to the reference manual for the fine details before using this operation.

Using the definition facility is not so hard as we may seem to have made out. In practice one doesn't do it very often so you will inevitably have to remind yourself of the procedure by reading the reference manual whenever you do.

3.6.4 Find and Replace

The Find (and Replace) function is a very powerful and easy-to-use facility. If you click on Find under the Window button on the console a window will appear divided into two equal-sized subwindows. The top one, the *find window*, specifies what is to be found, and the bottom one, the *replace window*, what the replacement should be. The required operation is then chosen from the menu under the Find button at the head of the window. The choices are Find, Replace, Replace&Find and Replace All. For the Find operation it is, of course, not necessary to specify a replacement. Replace All is a global find-and-replace facility. Use it with great care! Not only is it difficult to undo, it may also cause the system to loop indefinitely.

All find or find-and-replace actions take place in the edit document in which the target selection was last set and begin at the last such position of the target selection. If the message "Not found" appears in the console then check the position of the target selection. Sometimes in the processing of specifying the action it may be convenient to reset the target selection to another position in the edit document or even in another document. If you have done so, remember to set the target to the position you want the search to begin from before initiating the action. (Resetting the target selection to within the find-and-replace window is inevitable during the course of specifying a find or find-and-replace action but has no effect on the search process.)

The fact that the find and replace windows are just like any other document window, whilst having the one slight disadvantage mentioned above, is otherwise a major contribution to the ease of use of the facility. Everything we have explained above about editing expressions can be carried out in these windows. There are no limitations peculiar to find and replace.

There is one difference. Whenever you create a place holder in the find or replace window then it is numbered. If you create a place holder by copying a numbered place holder then the number is copied too, otherwise Math∫pad creates a new number. For instance it is possible to create the expression $\mathbf{E1} - \mathbf{E1}$ in the find window. If you create the expression in the normal way Math∫pad will create $\mathbf{E1} - \mathbf{E2}$. Now you can set the target to $\mathbf{E2}$ and the source to $\mathbf{E1}$ and copy the latter to the former thus creating the desired expression. Searching for $\mathbf{E1} - \mathbf{E1}$ in your document will find all occurrences of a repeated mathematical expression where the two instances are separated by a minus operator. (Spacing is ignored in this process.) If you also enter 0 in the replace window (and command the replacement to be made) the expression will be simplified according to the usual rule of arithmetic.

The find-and-replace facility can in this way be used to make complicated mathematical transformations to an expression. It is particularly useful when a large subexpression needs to be copied several times. (A so-called "complification" step in a proof. The use of find-and-replace in simplification steps, such as reducing $\mathbf{E1}-\mathbf{E1}$ to 0, is not so useful since it is so much quicker and easier to do by more direct means.) It is also possible to create and save a database of such transformations, although the mechanism to do so is quite primitive. Operations to do so can be found under the Stack button of the find-and-replace window. For further details see chapter 10.

You can also use find-and-replace to search for unfilled place holders. A "Delete" operation in the find (or replace) window executed on a numbered place holder will remove the number leaving behind an unnumbered place holder of the same type. Such unnumbered place holders match with *unfilled* place holders in the document to be searched. For example, entering a display place holder in the find window will automatically enter the numbered place holder "**D1**". The find operation will then seek all displayed expressions, including displayed expressions in which the place holder is empty. Selecting the display place holder and performing delete will cause "**D1**" to be replaced by "**D**". The find operation will then seek just those displays in which the place holder is unfilled.

Unnumbered place holders in the replace window have a similar function: the replacement is an empty place holder of the chosen type. For example, if $\mathbf{E1}-\mathbf{E2}$ is entered in the find window, and $\mathbf{E1}-\mathbf{E}$ in the replace window, then an expression such as $x-y$ will be replaced by $x-\mathbf{E}$.

The find-and-replace facility can, of course, be used for much more mundane purposes. For straight text an Emacs look-alike search and replace facility is also provided.

A very pleasant aspect of Math∫pad's find-and-replace is that you can always interrupt any sequence of such actions at any time — without taking any special steps. If you are like us you will often find that during a global edit action you spot other items in your document that should be altered, sometimes as a consequence, sometimes not. If so then just go ahead and make the changes! The find-and-replace can always be resumed when you are ready, and you don't have to key in some magic sequence of control characters to interrupt it.

3.7 Conclusion

This completes the tutorial. Best of luck with your further efforts at using Math∫pad. We'd be pleased to hear of your experiences.

Part II

Reference Manual

Chapter 4

Basics

4.1 Introduction

Math∫pad has been written to run in the X11 window environment and we have assumed when writing this manual that the user is familiar with the basics of this environment. In this chapter we introduce the basic terminology used throughout the manual and describe briefly each type of window.

This manual is itself a Math∫pad document and thus can be read either on screen or in the hard-copy version. In the on-screen version, references to other chapters or sections take the form "file:filename". (In the hard copy version these references are replaced by numbers.) To view such a chapter or section you should load the named file into a document window.

Math∫pad can be configured to your own wishes. For example, most commands that occur in menus can be executed via the keyboard using a combination of control and alphanumeric keys, but the particular combination that is used can easily be altered (see chapter 14 for details). The names of menu items can also be easily altered (for example to a different language). The discussion in this manual relates to the system as distributed. Consult your local expert for details of modifications that have been made. (If you do make such modifications our advice is to update the manual at the same time.)

The following font styles are used to highlight different items.

Italic First, or defining, occurrence of important terms.

Sans serif	Options. These can be buttons at the head of a window or an item in a pin-up menu.
Bold	Keyboard commands or a sequence of keyboard commands.
`Type writer`	File names or LaTeX macros.

4.2 Terminology

4.2.1 The Mouse

Mathpad uses three mouse buttons.[1] We refer to these buttons as the *near (mouse) button*, the *middle (mouse) button*, and the *far (mouse) button*. If no mention is made of a specific mouse button then which button is used is not significant.

If you are right-handed then the near button will most probably be the leftmost button on your mouse, and the far button will be the rightmost button. If you are left-handed it may be the other way round. You can easily determine which is the case by pointing the mouse at any word in this text and double clicking with the leftmost mouse button. If the word that you pointed at is then displayed in inverse video mode then the leftmost mouse button is indeed the "near" button.

Several different mouse operations are used. The two basic ones are the *click* and the *drag*. A *click* is a quick press and release of a mouse button and a *drag* is the sequence: depress a mouse button, move the mouse and then release the mouse button. The other mouse operations are formed from sequences of these two operations. Thus a double click is a click immediately followed by another click, and the double drag is a drag followed by a click and another drag. In a similar manner other operations such as triple clicks, triple drags, quadruple clicks, quadruple drags and so on can be made. A description of the correct technique for making multiple clicks can be found in section 3.4.1.

4.2.2 Windows

Mathpad uses several types of window. What you are reading now (if you are doing so at your computer) is in a *document window*. Each window has a *header* which enables

[1] If your mouse has fewer than three buttons then it is usually possible to simulate the third button and still use Mathpad.

you to identify the type of window. The header of a document window, for example, consists of “MathSfile :” followed by the name of the document. Below the header is a sequence of (window) *buttons*. Examples of those on a document window are Load, Save and Rename. In this manual we use the sans serif font to indicate window buttons. Below the window buttons is the *windowpane*. (Actually, the find & replace window has two windowpanes.)

The different types of window are:

- The *console*.
- A *document*.
- The *find & replace* window.
- The *buffer*.
- A *stencil* and a *define* window for a stencil.
- The *default* window.
- The *symbol* window.
- The *file selector*.

The console is headed by the text “MathSpad” followed by the version number and the project name. (More on projects later.) It is always visible unless you have turned it into an icon. For all the other window types (except a document, which has already been described) the type appears in the header. Document windows, the find & replace window, and the buffer are referred to collectively as *edit* windows, since they share the property that all edit operations can be applied to their contents.

Stencils and the symbol window are so-called “palettes”. A *palette* is a windowpane containing a number of items each of which can be inserted into a document. In the case of the symbol window the items are symbols (typically, but not necessarily, mathematical), and in the case of stencils the items are templates for user-defined structures.

During a session, an unlimited number of documents and/or palettes may be open. For each of the other types of window there is always at most one open. In particular, it is not possible to “define” two stencils simultaneously: two or more stencils may be open but only one define window.

A brief description of each type of window is given in section 4.5. For more detailed information on each type of window consult first the contents or the index.

4.2.3 Selections

Text editors invariably make use of a *cursor* to indicate the position in a text at which input is entered or changes are made. Math∫pad, being a combination of a text editor and a structure editor, makes use of both a cursor and a so-called *target selection*. A cursor may be set simultaneously in each of the define window, the default window and the file selector. In such windows, the cursor is indicated by an upwards pointing triangle, and it has its usual function. The target selection is set in *exactly one* of the edit windows (a document window, the buffer or the find & replace window), and is displayed in inverse video mode. (The background is black and the text is white.) The target selection may be empty (which is different from not being set!) in which case we sometimes refer to it as the *edit cursor*. The edit cursor is indicated by a black diamond.

The target selection is where changes to the document occur (for example, by entering input from the keyboard). The term selected by the target selection is called the *target*, or the *target term* if we particularly want to emphasise the distinction from the target selection. Several functions use the target selection or target term as an argument. There is also a *source selection* and a *source (term)*, used by functions with two arguments. The source selection is underlined with a thick line on a black-and-white monitor, and coloured blue on a colour monitor. Some functions use, in addition, an *argument selection*. The term selected is called the *argument*. The argument selection is indicated by a thin underline on a black-and-white monitor, and a purple background on a colour monitor. These three selections can be made with the mouse, where the near button is used for the target selection, the middle button is used for the source selection and the far button for the argument selection.

If two of the selections overlap then the colour changes at the point of overlap. The overlap of a target and source selection is coloured yellow. That of a source and argument selection is coloured light blue and that of a target and argument selection is coloured green. Finally, if all selections overlap the colour is red.

From now on, whenever we refer to Math∫pad-specific commands we use the terminology *target*, *source* and *argument*. When we refer to X11-specific commands we use the terms *near*, *middle* and *far* (mouse) buttons.

4.2.4 Command Keys

Math∫pad uses the shift, control and meta keys from the keyboard to bind functions to key combinations. Every key can be used in combination with these three *command* *keys*. The command keys are used in the normal way: first hold down the command key, then depress and release the appropriate alpha-numeric key. Use of the command keys will be indicated by a prefix: either "**C-**" for the control key, "**M-**" for the meta key, "**S-**" for the shift key and "**CM-**" for both the control and meta keys. For example, **C-U** means, hold down the control key and then enter **U** from the keyboard. Use of the shift key as a "command" key is only significant in combination with the function keys (labelled "F1", "F2" *etc.* on the keyboard). In particular, there is no difference between, for example, **C-u** and **C-U**, both being equally acceptable.

The key used for the meta key is one of the modifier keys on the keyboard. On a Sun keyboard it is the ◇ key while on other keyboards one of the Alt keys is usually used. When a keyboard doesn't have a meta key, it is possible to simulate it with an unused key. This is a feature of X11 and the xmodmap program can be used to check which key is defined as the meta key (mod1) and to define a new meta key.

4.2.5 Projects

When using the Math∫pad system one is usually working on a project. A *project* consists of a number of documents, a number of stencils and the default settings for the project. The project name is displayed after the version number in the heading of the console. Details of projects are stored by the system in *project files*. The name of a project file is the name of the project followed by the extension "`mpj`". Project files are stored in the Math∫pad home directory as given by the system variable `MATHPADHOME`.

The use of projects enables you to organise your work around particular tasks. You can switch from one project to another or discontinue your work in the knowledge that you can resume the project at a later stage at the point at which you left off. In order to achieve the latter, the project file also stores information about the position of windows and their size. This includes information about auxiliary windows such as the buffer and the default window irrespective of whether they are opened or not, so that when they are opened they are positioned at the point at which *you* want them to appear instead of some fixed position dictated by system programmers. It does not, unfortunately, store information about the position of cursors, so this information is lost when exiting from a project and you are not able to resume *exactly* at the point where you left off.

Project files are ASCII files including comments on the information they contain so that you may edit them with any text editor instead of via the Math∫pad system. (This may be useful if, for example, you are very particular about the exact position of one of the windows. Positioning or resizing a window by dragging is not as precise as supplying the desired coordinates. In most circumstances, however, the information stored by Math∫pad is perfectly adequate.)

File Types

Various types of file are created or consulted. The different types are identified by a three letter *extension* to the name of the file. The extensions (all of which are preceded by a period in the standard way) and the file type to which they correspond are enumerated in table 4.1.

extension	file type
`mpd`	document
`mpf`	font selection file
`mpj`	project
`mpk`	keyboard bindings
`mpm`	font size information
`mpr`	find & replace actions
`mps`	stencil
`mpt`	LaTeX macros

Table 4.1: File Types.

The system uses the `mpk` file specified in the default window to determine the effect of command keys like the control key. It uses `mpt` files to associate fonts and symbols in the symbol palette with LaTeX macros and `mpm` files to relate size and font information. These ASCII files may be altered to suit your own requirements. See chapter 14 for further details.

4.3 Basic Techniques

We recommend that you read the welcome chapter and work through the tutorial on the first occasion that you start up Math∫pad. To do so, execute the command `mathspad manual &`. (If that doesn't work try `mathpad`, `Mathpad` or `MathSpad` instead of `mathspad`. If none of these work consult the person who installed the system about where it is located and under which name.) The console, shown in figure 4.1, will appear on your screen as well as a welcoming document.

Figure 4.1: The Console

Usually, Math∫pad is given the name of a project to open — "manual" is the name of a project supplied with the system. The argument may also be the name of a document or a stencil. (NB. The latter two options have not yet been implemented.) If no argument is given then a default project is opened. If you are a newcomer to the system — to the extent that the most you have done with the system is to read the welcome and work through the tutorial — and you execute the command `mathspad` (thus without an argument) then this is what happens.

Unlike other editors, Math∫pad does not use the directory in which it's started. Instead, a default user directory is used which is specified in the file `~/.mpdefaults`. This file contains the names of the directories where the documents, stencils and LaTeX files are found and written to by default.

If the file with the defaults does not exist (as will be the case if you have never used Math∫pad before), or the specified directories are not available, Math∫pad will create these directories on demand. By default the directories `~/mathspad/document`, `~/mathspad/stencil` and `~/mathspad/latex` are used for the different files. (These defaults can be changed in the default window: see chapter 13 for further details.) After creating these directories (with your permission!) the console will appear on your screen together with a document window. You are ready to begin creating your first Math∫pad document.

4.3.1 Scrolling

At the side or top of some windows is a *scrollbar*. This is useful when the window is not large enough to display all the information simultaneously. A scrollbar is recognisable by a black bar, which indicates the position of the displayed information relative to the total document. The black bar can be moved by clicking or dragging whilst the mouse is positioned in the scrollbar. You can see when the mouse is in the scrollbar because its shape changes from a single-headed to a double-headed arrow. (This may depend on your window manager, or on the way you yourself have configured the system.)

Horizontal scrollbars are used much less frequently than vertical scrollbars. The following description refers to a vertical scrollbar but the same principles apply to a horizontal scrollbar — replace "down" by "to the right", "up" by "to the left", *etc.*

The near mouse button moves the bar down. To move a particular line in a document to the top of the window, click the near mouse button whilst the mouse pointer is at a position in the scrollbar alongside the current position of that line.

The far mouse button moves the bar up. Its effect is the precise inverse to that of the near button. Thus, if you click in the scrollbar with the far button and, *without moving the mouse*, immediately follow this by clicking with the near button, then the total effect is as if you had done nothing. The same applies if you reverse the order: first click with the near button and, without moving the mouse, follow this by clicking with the far button. In particular, if you want to move back one window in a document then click the far button at the lowest point in the scrollbar. If you just want to move back one line then click the far button with the mouse positioned in the scrollbar pointing at the second line of text in the window. The further down in the scrollbar, the further back the window will scroll.

The middle button is used to move the top of the bar to a position relative to the document as a whole. To understand what happens you have to see the black bar as representing the displayed window, and the whole scrollbar as representing the whole of your document. If, for example, a window shows the whole of a document, the scrollbar will be entirely black. The smaller a window is relative to the document as a whole, the smaller the black bar will be relative to the total length of the scrollbar. Moreover, the position of the black bar in the scrollbar indicates the relative position of the window to the document as a whole: if the window shows the very beginning of a document then there will be no white space above the black bar; if the window is in the middle of the document there will be (roughly) equal amounts of white space above and below it. By dragging with the middle button you can thus scroll the text dynamically. Dragging downwards moves forward in the text, and dragging upwards moves backwards.

4.3.2 Pin-Up Menus

Pin-up menus are used for showing a number of options. Some examples of menus are shown in figure 4.2. Each has a *header*, which is the name of the button to which

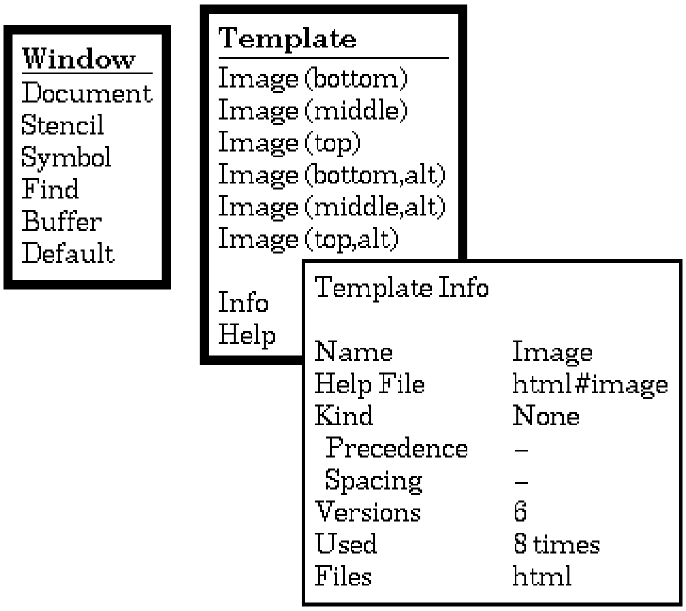


Figure 4.2: Some Pin-Up Menus

they belong or "Template" for menus associated with the templates in a stencil, and are separated from the menu items by an underline.

A menu will appear when you click on a button or a template with the far mouse button; you can then make a selection, or the menu can be pinned up. Using a mouse button other than the far mouse button will select the first item of the menu.

To select an item move the mouse without releasing the mouse button until the required item is highlighted. Then release the mouse button. Dragging the mouse from a position within to a position without the boundary of the menu and then releasing the mouse button will cause the menu to disappear without any action having been performed.

Some menus have sub-menus, indicated by a triangle pointing to the right. By clicking on or dragging over that option, the sub-menu will appear and the selection can be made. If you move back to the main menu, the sub-menu will disappear.

In some situations, you might want to use the same menu repeatedly. For this purpose, you can pin the menu up by selecting the menu's header. On a colour monitor the menu will acquire a grey boarder indicating that it is "securely" pinned up. Once pinned up, items from the menu can be selected using any of the mouse buttons; the menu will not then disappear after the selection has been made, nor if the mouse is dragged outside the boundary of the menu. While in the pinned-up state the menu can be dragged to another position on the screen in the usual way that X11 windows are repositioned: point the mouse at any edge whilst holding down the near button and then drag. To remove a pinned-up menu select the header of the menu.

The indication of whether or not the menu is "pinned up" depends on your window manager. Also, depending on your window manager, a "pinned-up" menu may disappear if the parent window is closed.

4.4 Pop-Ups

Pop-ups are warning messages or requests for information that "pop up" on your screen. An example is the message that appears when you quit the system without having saved a modified document: you are warned that the document hasn't been saved and asked whether you wish to do so. Unlike the messages that appear in the console, pop-ups have to be dealt with before any further progress can be made. (If you try to proceed without responding to a pop-up then you will be beeped and a message will appear in the console.)

The design policy is to have as few pop-ups as possible on the grounds that users know better than the system what they are doing. Novice users may, however, appreciate the support that warnings can give. There is thus a default option specifying the level of support that is desired. (This feature has not yet been implemented and the current system is geared to proficient users.)

4.5 Windows

In this section we give a brief description of each of the different window types. Complete descriptions are given in subsequent chapters.

4.5.1 The Console

When Math∫pad is started, the user is presented with the interface shown in figure 4.1. We call this window the *console*. In the console there is a row of buttons[2] whose purpose is to open other windows or to apply functions to selections. There is also a small strip below the buttons where messages are displayed. The actions associated with the buttons are listed in chapter 6 and described in detail elsewhere in the manual.

4.5.2 Document Window

A document window shows (part of) a document, the name of that document being shown in the header of the window. A user can have several documents open at once so that different documents can be combined or parts copied from one to another. A document window can also be used as a scratch pad to hold temporary structures and parts.

Documents are stored in files whose name is composed from the name of the document and the extension "`mpd`", "mpd" being an abbreviation for "Math∫pad document". For example, the document you are reading now is stored in the file `Basics.mpd`.

Backups of documents are made at regular intervals (specified in the defaults window). The name of the backup file of, for example, `mydoc.mpd` is `#mydoc#.mpd`. The directory in which the backup is stored is normally the directory from which the document was loaded. If it is not possible to use that directory then the `/tmp` directory is used.

A document may be loaded in *view mode* as indicated by the word "view" in parentheses following the name of the document in the header. There are two occasions when this occurs: when sections of the manual are loaded using the help facility, and when a document that is already loaded is loaded for a second time. (The user has thus no control over when view mode is used.) The meaning of view mode is that no backups of the document will be made and if the document is modified then that this is the case will not be indicated in the header.

More information about documents can be found in chapter 5.

[2]In a later release, the user will be able to adjust the layout of the console by removing or adding functions.

4.5.3 Stencils

A *stencil* is a file defining notational conventions and constructions. It is the software equivalent of the stencils used, for example, to sketch the apparatus in a chemical experiment. Each stencil comprises descriptions of a number of *templates*. Each template has one or more *versions* and (possibly empty) associated *help file*. The description of each template version includes a description of its on-screen format and the output it should generate.

Stencils are stored in files whose name is composed from the name of the stencil and the extension "`mps`", "mps" being an abbreviation for "Math∫pad stencil". Stencils are opened by selecting the Stencil option in the Window menu.

The window used to access an existing stencil or to define a new stencil is called the *stencil palette*. An example of a stencil palette is shown in figure 5.2. There is no restriction on the number of stencil palettes that can be open at any one time. The use of stencils is discussed in more detail in chapter 8 and their creation in chapter 11 and chapter 12. There is a built-in mechanism to help the user specify LaTeX output, but there is no obligation to generate LaTeX. Help files are normal Math∫pad documents.

4.5.4 Define Window

Users can customise Math∫pad by defining their own stencils and templates. The definition of a new template or the modification of an existing template takes place in the *define window*, an example of which is shown in figure 11.1. A define window is opened by clicking in the Define button of the appropriate stencil palette. The creation and update of stencils is described more fully in chapter 11 and chapter 12.

4.5.5 The Buffer

The *buffer* is used by the system to store parts of a document that have been deleted as a result of an edit command. Its primary purpose is to assist recovery from inadvertent edit actions. The window can be used like a normal edit window, except that its contents cannot be saved; it may thus be usefully used for scratch work. It is opened by selecting the Buffer option in the Window menu.

There are two buttons at the head of the Buffer window — Kill and Done. The Kill button clears the contents of the buffer. (This cannot be undone!) The Done button closes the window. Items are continuously added to the buffer whether or not it is open, and closing it does not destroy its contents. The contents of the buffer are, however, lost between Math∫pad sessions and when changing project.

4.5.6 Find & Replace Window

The *find & replace window* is used to search for or (repeatedly) replace an item. For everyday textual search and/or replacement (seeking and possibly replacing parts of words) the Emacs-like facilities are often more convenient. The find & replace window is most useful for structured items. There is a primitive mechanism for recording find & replace actions, so it may also be used to record and apply mathematical laws. The window is opened by selecting the Find option of the Window menu. The window contains two parts both of which can be used like a normal Math∫pad document window. The upper part contains the construction to search for, whilst the lower part contains the replacement (if necessary). Full details of its use can be found in chapter 10.

4.5.7 Symbol Palette

Many documents, especially mathematical documents and texts in foreign languages, contain special symbols. To make these symbols easily accessible to the user, the *symbol palette* is available. It can be opened by selecting the Symbol option in the Window menu. The palette displays a subset of the available symbols, as shown in figure 5.3. The required symbol can be selected from the palette by clicking on it with the mouse. More information can be found in section 5.3.

4.5.8 Defaults Window

In the *defaults window*, as shown in figure 13.1, some default settings of Math∫pad can be changed. These settings affect the on-screen layout, the LaTeX output and specify the directories used by the editor. The options available and how to change them is explained in chapter 13.

4.5.9 File Selector

The *file selector*, shown in figure 5.4, is used to search for and select files. The file selector appears as a result of requesting a document or stencil to be loaded, renamed or saved, by requesting a document to be converted to LaTeX, or by requesting a document or (ASCII) file to be included in an existing document. It uses a *mask* to select and display all the files in a certain *base directory*. When used to load, rename or save a stencil the mask is set to `*.mps` indicating that only `mps` files are to be shown; when used to load, rename or save a document the mask is set `*.mpd`; finally, when used to convert the output to LaTeX the mask is set to `*.tex`. The mask can be edited so that it is possible to use the file selector as a general-purpose tool. For further information on the file selector see section 5.4.

Chapter 5

More on Windows

This chapter provides further information on document windows, stencil palettes, the symbol palette and the file selector. In the case of documents and stencils the information is confined to a description of the available buttons. The creation of documents is discussed in chapter 7 through chapter 10 and the creation of stencils is discussed in chapter 11 and chapter 12.

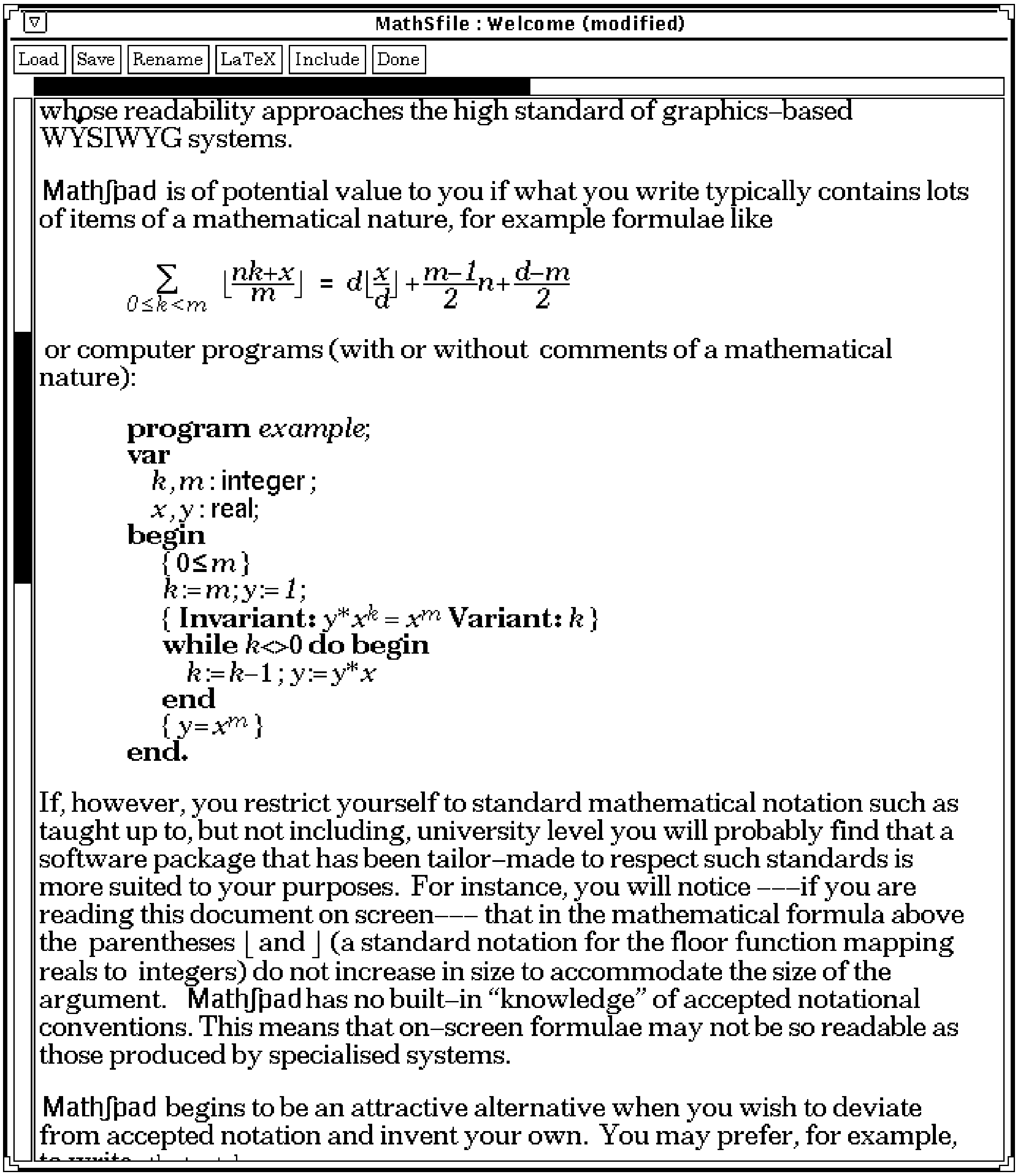
5.1 Document Window

To edit a document, a document window, as shown in figure 5.1, must be open. If there is no document window available, one can be opened by selecting Document from the menu under the Window button in the console. Initially, the window will be empty and it, and its icon, will have the name noname?? where ?? is a number that indicates how many edit windows have been opened. When an existing document is loaded into the window, the name will automatically change to that of the filename. The window name can also be changed by saving a document to a different filename or with the Rename command. To create a new document a new window should be opened.

Open documents are regularly saved to a backup file at intervals specified in the default window. The name of the backup file corresponding to `doc.mpd` is `#doc#.mpd`.

At the top of a document window are some buttons. The commands associated with these buttons are explained below.

Load Open the file selector in order to select a document file to be loaded. The



whose readability approaches the high standard of graphics–based WYSIWYG systems.

Math∫pad is of potential value to you if what you write typically contains lots of items of a mathematical nature, for example formulae like

$$\sum_{0\le k<m} \lfloor \frac{nk+x}{m} \rfloor = d\lfloor\frac{x}{d}\rfloor + \frac{m-1}{2}n + \frac{d-m}{2}$$

or computer programs (with or without comments of a mathematical nature):

```
program example;
var
  k,m : integer;
  x,y : real;
begin
   {0≤m}
   k:=m;y:=1;
   { Invariant: y*x^k = x^m Variant: k }
   while k<>0 do begin
      k:=k–1 ; y:=y*x
   end
   {y=x^m}
end.
```

If, however, you restrict yourself to standard mathematical notation such as taught up to, but not including, university level you will probably find that a software package that has been tailor–made to respect such standards is more suited to your purposes. For instance, you will notice –––if you are reading this document on screen––– that in the mathematical formula above the parentheses ⌊ and ⌋ (a standard notation for the floor function mapping reals to integers) do not increase in size to accommodate the size of the argument. Math∫pad has no built–in "knowledge" of accepted notational conventions. This means that on–screen formulae may not be so readable as those produced by specialised systems.

Math∫pad begins to be an attractive alternative when you wish to deviate from accepted notation and invent your own. You may prefer, for example,

Figure 5.1: The Document Window

name of the window and its icon are changed to that of the loaded file. If there was previously a document in the window, this is removed and saved in a backup file. If the selected file is already loaded in another document window the file is loaded in view mode and a warning is given.

Save Save the document that is being edited. The file selector appears showing to which file the document will be written and offering the possibility to save the document in another file. (Doing the latter will also change the name of the document window.) If the save fails for some reason, a message will appear in the console.

Rename Rename the document window and its related icon. Note that renaming does not affect the name of the `mpd` file. Thus, if a file is loaded, and Rename is executed followed by Save, two copies of the loaded file will be obtained, one with the original name and one with the new name.

Output Convert the document to LaTeX. The default name of the LaTeX file is the name of the document with the extension "`tex`" but the name can be changed via the file selector that appears.

Include Include a document at the position of the cursor. The document may be a Math∫pad document or an ascii file.

Done Close the window. If the document has not been modified since it was last saved, the document and window will be immediately removed. Otherwise the document window will have "**(modified)**" after its name in the header and the user will be asked by means of a pop-up whether the file should be saved or the action aborted. Warning: the "**(modified)**" indication is sometimes unreliable if a change has been made to a document as the side effect of a change to a stencil.

5.2 Stencils

A ***stencil palette***, as shown in figure 5.2, can be opened by clicking on the Stencil entry in the menu under Window on the console. An empty palette will appear. To load an existing stencil file into the palette click on Load and then choose the required stencil using the file selector window.

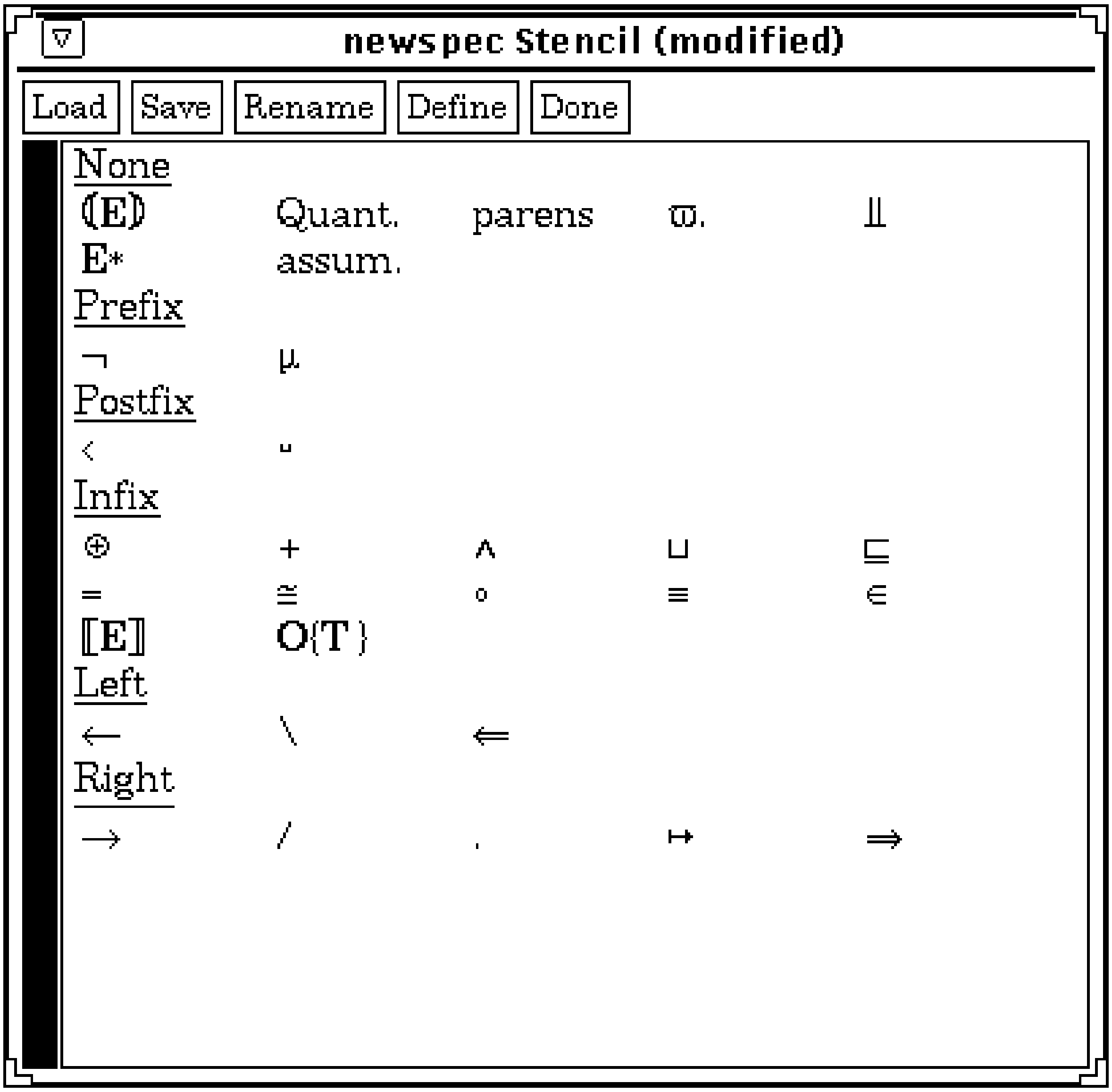


Figure 5.2: A Stencil Palette

It is possible to open any number of stencils including several copies of the same stencil. The templates are displayed in columns and when the window is resized the number of columns is automatically adjusted to fit the new size. The scrollbar makes it possible to see different templates. If the stencil is particularly big having several windows on the same stencil can obviate the need to use the scrollbar.

At the top of each stencil window are five buttons which are used as follows:

Load Remove the current contents of the window and load the stencil from a specified file. The file selector appears in order to facilitate the selection. Loading two files with the same name from different directories is automatically detected and signalled in the header of the file loaded second by a number between angle brackets. For example, the header File <1> Stencil means that a stencil file called File has already been loaded from another directory.

Save Save the stencil in a specified file. The file selector appears in order to facilitate the selection.

Rename Rename the stencil window. Note that the name of the file is not altered. Thus a rename followed by Save will have the effect of creating two (possibly different) copies of the file.

Define Open the define window to create new templates or to change existing templates. The define window is linked to the stencil and new templates will be added there. If the define window is already open, its contents will not be changed but the stencil to which it is linked will be reset. See section 11.2 for more information on this window.

Done Close the stencil (and the define window if also open).

In the case of Load, Save and Rename the file selector automatically appears in order to facilitate the selection of the file name. The file selector uses the default stencil directory as base directory for Load. For Save and Rename the directory containing the already loaded stencil is used.

Templates are divided into separate groups according to their kind. These groups are labelled "None", "Prefix", "Postfix", "Infix", "Left" and "Right". Within a group the templates can be rearranged by moving them left or right using the **M-left** and **M-right** keys. The versions of a template are rearranged via the define window. See section 11.3.7 for more information.

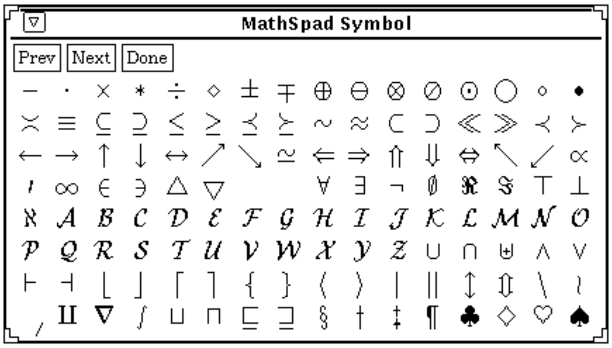


Figure 5.3: A Symbol Window

To select a template simply click on it and the first version of the template will be entered at the position of the cursor. If another version of the template is needed, the far mouse button must be used to select the template. The menu that appears lists the different versions available and the required version can be chosen.

5.3 Special Symbols

To use special symbols a ***symbol palette*** should be opened. This is done by clicking on the Symbol menu item under Window on the console. The symbol palette has a number of pages; by opening several copies of the palette it is possible to view a selection of the pages simultaneously. It is also possible to resize each opened symbol window independently. This is useful because of the different sizes of the symbols on different pages of the symbol palette. When a window is resized, the symbols will be placed so that they are readable and as widely spaced as possible.

A symbol window, as shown in figure 5.3, contains three buttons and at most 128 special symbols. At the top of the window, the following buttons are available:

Prev Go to the previous page in the palette.

Next Go to the next page in the palette.

Done Close the palette.

The buttons Prev and Next are used to select different pages. Which pages are available depends on your local installation of Math∫pad. It is always possible to add more pages (up to a certain limit) and/or more symbols. See chapter 14 for details.

A symbol can be selected by clicking on it with the mouse. It's possible to change the selection by dragging the selection to another symbol. The selected symbol is highlighted and will be handled like a normal keystroke. The palette with special symbols should be seen as an extension of the keyboard. In some cases the selected symbol will not be used because the symbol is not permitted in the string — for example, in a file name.

When LaTeX output is created, every occurrence of a symbol will be replaced by the LaTeX macro that is loaded when the editor is started. Mathspad will automatically select the correct LaTeX mode and mode braces (the symbol "$" before and after an expression) are usually not needed. These macros can be changed if necessary and you can add your own macro to a symbol in a special file (with extension `mpt`). See chapter 14 for more details.

5.4 File Selector

The file selector is invoked in several ways: by requesting a project to be saved, a document or stencil to be renamed, loaded or saved, by requesting a document to be output, or by requesting a document or (ascii) file to be included in an existing document. At the head of the file selector there are three fields: the action to be taken (e.g. "Load document" or "Make output file"), a selection and a mask. The selection is the name of either a directory — recognisable by a "/" at the end of the name — or a file. Both the file name and mask can be edited using the text editing commands described in chapter 7.

The *base directory* is the directory containing the file named in the "Selection" entry, if that entry is indeed the name of a file, or the directory itself if the entry is a directory.

Beneath the header are two windowpanes labelled "Directories" and "Files". The "Directories" windowpane contains the names of all the subdirectories of the base directory, and the "Files" windowpane displays the names of all the files in the base directory matching the mask. The rule for matching a name against the mask is that the symbol "*" matches with any sequence of characters, the symbol "?'' matches any sequence of exactly one character, and all other characters have their literal meaning. The entry ".." in the "Directories" windowpane is the parent directory of the base directory (in accordance with the usual convention). Both of these windowpanes have a vertical scrollbar. Any entry in either of these windowpanes can be selected by clicking on it with the mouse.

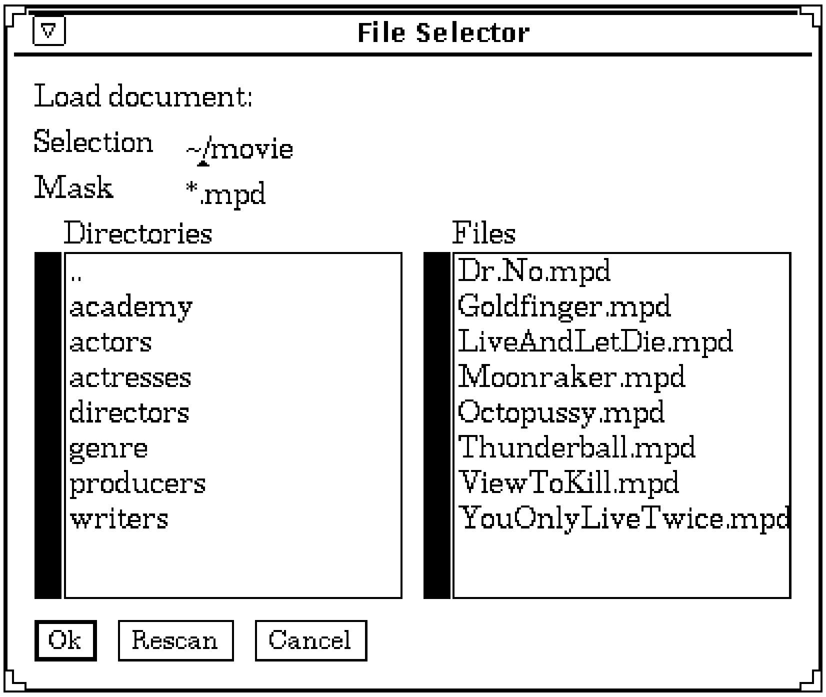


Figure 5.4: The File Selector

Selecting a directory causes the directory to be entered in the "Selection" field. A double click on a directory will also cause the Directories and Files windowpanes to be updated. (The name entered if ".." is selected is the name of the parent directory.) Selecting a file causes the filename to be entered in the "Selection" field; no change occurs in the two windowpanes.

Below the two windowpanes are three buttons: Ok, Rescan and Cancel. Clicking on Cancel causes the file selector to disappear without taking any action. Clicking on Ok causes the action for which the file selector was invoked to be carried out. Rescan is used to update the "Directories" and "Files" windowpanes after editing the mask or "Selection" field.

A shortcut to selecting a file and clicking on Ok is to double click on the name of the required file in the "Files" windowpane.

During execution of the invoked action the file selector remains visible and the mouse pointer changes shape. On completion, the file selector disappears.

The set of symbols that can be entered in the file name or mask is limited by the usual conventions on the names of files and directories. That is, all alphanumeric characters may be used together with the characters "/" (the separator between directories and between a directory and file name), "~" (the user's home directory) and the character sequence ".." (the parent directory). An invalid or non-existent file name (constructed by editing) will be truncated to the largest valid prefix when the Rescan operation is invoked. If Ok is invoked, however, a pop-up appears indicating the nature of the error.

Chapter 6

Console Commands

This chapter summarises all commands executed via the console (see figure 4.1). The most frequently used console commands can also be executed via the keyboard; the key-sequences needed to do so are listed. The sections correspond to each of the buttons on the console. Under each button is a menu. To execute a command, select the appropriate menu item and then click. In each case the default item chosen by clicking with the near mouse button on a console button is the first menu item.

There is no undo command. In most cases, however, commands have obvious inverses and can be manually undone. Where the inverse is not obvious, but does exist, an explanation of how this is done is given. When undoing a command may be problematic a warning is given. Execution of such commands should, of course, be done with care!

6.1 Window

The Window button is used to open windows. The different types of window are listed below.

Document Open a window in which to input a new or edit an existing document. The number of documents that are simultaneously open is, in principle, unlimited. See section 5.1 for further information about documents.

Stencil Open a stencil palette in order to use and/or create the templates of a stencil. A click with the mouse on Stencil opens an empty stencil palette. A description of how to load an existing stencil into the window is given in section 5.2. Information on creating new stencils can be found in chapter 11 and chapter 12.

Symbol Open a symbol palette to access special symbols. The symbol palette contains several pages of symbols, and several copies of the palette may be open at any one time. For more information see section 5.3.

Find Open the Find & Replace window in order to find and (possibly) replace structured expressions. Use of this window is described in chapter 10.

Buffer Open the buffer window. For further information see section 4.5.5

Default Open the default window. See chapter 13 for further information.

There are no restrictions on opening documents. In particular, the same document may be opened several times. This is a useful option because of the convenience of being able to view different sections of a document at the same time. The current implementation is, however, inadequate. The different windows are not coupled to each other, each containing its own copy of the original document. Edit actions in one window will thus not change the copy in another window. Extreme care should therefore be taken when opening different windows on the same document.

There are also no restrictions on opening stencils and the same stencil may be opened several times. In contrast to documents, however, alterations to stencils take immediate effect in all windows displaying the stencil.

None of the above commands can be executed via the keyboard.

6.2 EditOp

The EditOp button gives access to the most frequently used general-purpose edit operations on structured expressions.

Insert Expr Insert a place holder for a mathematical expression.

Insert Disp Insert a place holder for a displayed expression.

Copy Replace the target term by a copy of the source term. Note that the original target term is placed in the buffer and so can be restored from there if necessary.

Swap Interchange the target and source terms.

Delete Erase the target term and put it in the buffer.

Insert Id Insert a place holder for an identifier.

Insert Op Insert a place holder for a mathematical operator.

Insert Text Insert a place holder for text.

Insert Var Insert a place holder for a variable.

Note that the entries in this menu are ordered more according to frequency of use rather than being grouped by functionality.

Place holders (for text, expressions, operators, variables and identifiers) are most often entered into a document via the selection of a template, and thus *not* via the EditOp button. The need does arise, nevertheless, to insert a placeholder on its own and it is for this purpose that the various "Insert" options are available. For example, in the text "let x denote ..." an expression or identifier placeholder is necessary before the "x" can be entered. In addition, displayed items are always entered in this way. A placeholder for a displayed expression is entered by selecting the Insert Disp item. A placeholder for an in-line expression is entered by selecting the Insert Expr item, and so on. Inserting a text placeholder enables the use of a mixture of mathematics and prose in a mathematical expression.

Because of the frequency with which the first four of these operations are used, they are bound to both function keys and to **CM-**keyboard commands. The other operations are bound to **CM-**keyboard commands as given in table 6.1.

The Delete operation is executed from the keyboard by depressing the key labelled "Backspace", "Delete" or "Del" depending on your configuration. Its function is to delete the target term.

6.3 Structure

The Structure button provides editing operations that are typical of (but not necessarily confined to) mathematical calculations. With the exception of Apply and Rename each operation has an obvious inverse and, thus, can easily be undone. Undoing Apply is detailed below. Undoing Rename may be problematic; executing it should be done with care!

Command	Function key	Key combination
Insert Exp	**F5**	**CM-E**
Insert Disp	**F6**	**CM-P**
Copy	**F7**	**CM-C**
Swap	**F8**	**CM-S**
Insert Id		**CM-I**
Insert Op		**CM-O**
Insert Text		**CM-T**
Insert Var		**CM-V**

Table 6.1: Keys for basic edit operations.

Add/Rem () Insert or remove parentheses around an expression. Parentheses are inserted if there are none already, otherwise they are removed. Removal of parentheses is sometimes necessary because the system gets it wrong (after, say, a Distribute operation). Parentheses are not removed if the operator precedence requires their presence. To force their removal in such a case you may use Ungroup (described below). (Do so only if you know what you are doing!)

Reverse Reverse (i.e. commute) the order of the elements in a list of operands and associative infix operators.

Distribute Distribute the function determined by the source and argument selections over the expressions in the target selection.

Factorise The inverse of distribute. Factorise the function determined by the source and argument selections from all expressions in the target selection.

Apply Apply the function determined by the difference between the source and argument selections to the expression in the target selection. (The argument should therefore be a subexpression of the source expression.) For example, apply '$a+$' to '$b\times c$' giving '$a+b\times c$'. The resulting expression becomes the new target. To undo this operation, set the source selection to the original target selection and execute Copy.

Group Alter the structure of the expression by making the target a proper subexpression of the expression in which it occurs.

Ungroup Alter the structure of the expression by inserting the selected subexpression in the expression one level higher.

Rename Replace every occurrence of the target by the source in a range determined by the argument. Both the target and source selections should be identifiers. The operation may sometimes be undone by simply interchanging the source and target *selections* (see section 6.4) and repeating the operation, but this is not always the case.

Command	Function key	Key combination(s)
Add ()	**F4**	(,**M-(**, **M-9**
Remove ()		**C-(**, **C-9**
Reverse	**F9**	**CM-R**
Distribute		**CM-D**
Factorise		**CM-F**
Apply		**CM-A**
Group		**CM-G**
Ungroup		**CM-U**
Rename		**CM-N**

Table 6.2: Keys for edit operations on structures.

Note: Typing an opening parenthesis "(" will *insert* opening and closing parentheses around the target term provided the target term is an expression. If the target is text then the opening parenthesis is inserted in the normal way.

6.4 Selection

The Selection button gives access to functions that change the selections *but do not alter the edit document itself.*

Swap Interchange the target and source selections.

Clear Deselect a selection by clicking in the Clear button with the same mouse button.

Join Combine the selection indicated by the mouse button with the third selection. If the third selection is not set, the whole document will be selected.

Note: Interchanging the target and source selections is not the same as interchanging the target and source terms. The former has no effect on the document itself whereas the latter does.

There are no keyboard shortcuts for these commands.

6.5 Misc.

The Misc. button gives access to a miscellaneous collection of commands.

Word Wrap Insert and remove line breaks throughout the target term so that all text fits neatly within the window. There is no way of undoing this command but it has no effect on the output produced by LaTeX.

Output Convert the target term to its output value. The output is stored in the window system buffer and thus can be transported to (say) an email message or an Emacs document. This command has three options, namely LaTeX, Plain and Ascii. LaTeX generates LaTeX making use of the loaded stencils together with built-in knowledge. Plain generates plain LaTeX, that is LaTeX with no `\ms` commands and no braces `{}` around the operators. Ascii generates output using *literally* the information contained in the template definitions, and thus without special provision for LaTeX. The primary use of Output is to debug the definition of templates.

The buffer that is used is referred to as "the selection" on Sun systems. This terminology would, however, be confusing in this context and we refer to it instead as the *window port*.

Paste Insert the contents of the window port into the document. This command allows you to import information from other (non Math∫pad) windows, or system-generated information.

Goto Line Go to specified output line. Selection of this menu item will cause a request for a line number to appear in the console. Fill in the desired number and press return. To abort type **C-G**.

Toggle Dots Introduce "dots" delimiting text place holders. (See below for further explanation).

Display Left Move the display (selected as target) one tab stop to the left.

Display Right Move the display (selected as target) one tab stop to the right.

Spacing The three options increase, decrease or reset the elastic spacing in the target selection. ("Elastic spacing" is the spacing around mathematical operators and spacing specifically declared to be elastic in the template specification.)

Ispell This entry gives access to a menu for coupling Math∫pad to the Ispell spelling checking program. See below for a more detailed account of how this option is used.

Maple This entry gives access to a menu for coupling Math∫pad to Maple. The feature is experimental and is not documented.

LaTeX Last File Selecting this entry will cause the last file that has been converted to LaTeX to be processed by LaTeX. A Math∫pad document window will be automatically opened and the progress messages from LaTeX will be sent to the window. Note that the file must conform to the LaTeX format requirements; in particular, it should begin with the appropriate preamble. The automatically opened window is titled "LaTeX Shell". During processing of the file, this title is followed by "(running)"; once the process is complete, the message is changed to "(done)". This feature is experimental. Currently, the only means of interacting with the LaTeX program is to remove the Shell window: this has the effect of aborting the process. The document produced is a standard Math∫pad document. This can be exploited by, for example, using the Find facility to search for the location of errors in the processed document.

Keyboard commands corresponding to the Misc. menu items are shown in table 6.3. (Note: On a Sun keyboard the **F18** key is labelled "Paste".)

Elastic spacing can be increased by one unit at a time from the keyboard using either **C-+** or (more conveniently, since it doesn't involve using the shift key) **C-=**. Decreasing elastic spacing is effected with **C- –** (control-minus) and resetting with **C-0**.

The result of an Output command (executed via the console) is inserted into the window port from where it can be pasted into another window, for example an email message. In addition, Output and Paste are often used in combination to check the LaTeX output

Command	Key
Word Wrap	
Output	**F10**
Paste	**F3**, **F18**
Goto Line	**M-L**
Toggle Dots	**C-.**
Display Left	**CM-Left**
Display Right	**CM-Right**
Spacing	See below
Ispell	
Maple	
LaTeX Last File	

Table 6.3: Keyboard commands for miscellaneous menu items.

produced by a template: Select an instance of the template in a document, select Output , move the cursor to a suitable position (possibly in a scratch document or the buffer) and select Paste. The different output options can also be chosen using **C-U** as follows. (This possibility is normally used in combination with the **F10** function key but it is also possible to enter **C-U** followed by a number and then select the Output command with the mouse.)

C-U 1 F10
the default LaTeX produced by Math∫pad.

C-U 2 F10
the default LaTeX with no `\ms` commands and no braces `{}` around the operators.

C-U 3 F10
the output specified in the stencils literally interpreted (thus no provision for LaTeX)

The Toggle Dots option is used to introduce or remove dots delimiting text place holders. The introduction of such dots is sometimes necessary during preparation of a document in order to determine the extent of a text place holder or to ease moving the cursor to the beginning or end of a text place holder. The effect is limited to what is shown on the screen; neither the contents of the document nor the output or affected by this action.

6.6 Quit

The **Quit** button enables one to quit the current project and (possibly) start a new project.

Quit Quit the system. A warning is given of each document or stencil that has been modified but not saved. In addition, the project is saved if that is the chosen default. (This is not yet implemented.)

Fast Quit Quit the system immediately. No warnings are given and the project is not saved (independent of the chosen default). Use only when you are absolutely sure of what you are doing. The command cannot be undone.

New Project Change project. The file selector appears in order to facilitate selection of the new project. If the name supplied does not already exist then a new project file is created.

Save Project The status of all opened documents and stencils (i.e. their position on the screen, whether iconicised *etc.*) is saved in a project file which is specified by means of the file selector. Also saved is the status of all auxiliary windows such as the buffer and default window regardless of whether they are opened or not. The command can be undone if a new project name is entered every time the command is executed. This is of course impractical but the procedure can be followed if you are not completely confident of what you are doing.

For safety reasons, only the Save Project command is available via the keyboard. Typing **M-P** causes the file selector to appear enabling the project name to be selected.

Chapter 7

Text Editing

Plain (non-mathematical) text can be edited with a number of keyboard commands. The commands are based on those provided by Emacs and most of the usual functions are available. Some commands, however, behave slightly differently from those of most text editors, because of the possibility to use proportional fonts and because of the way the target selection is handled.

Each group of characters input with a *template* is called an *expression* and, for the purposes of text editing, is treated as if it were a single symbol. Moving the cursor character by character through the document will display such a group in inverse video mode as if it were a target selection. For example, move the cursor one character at a time along the sentence "In Math∫pad, there are three types of selection." (Use **C-F** or the right arrow key on your keyboard.) You will see that when the cursor reaches Math∫pad, the complete word "Math∫pad" is displayed in inverse video mode. Move one more symbol to the right and the cursor will be between Math∫pad and the comma. This is because Math∫pad was input with a template and is treated as a single character at the text level.

The effect of some operations depends on whether or not the target is set. It is important to remember that the cursor is considered to be indicating the position of an *empty* target.

With Math∫pad it is very easy to insert special characters into the text. These symbols are selected from a symbol palette and are then handled like normal characters in the editing actions as far as the screen version is concerned. However, if a text contains such symbols, the LaTeX code will be a little different. The symbols are bound to LaTeX macros and these macros will be used in the LaTeX code. The dollar signs (**$**)

are automatically added to switch to the correct mode and therefore do not need to be specially added to the normal text.

Arguments to keyboard commands

Caution: use this feature with care — even if the action you perform has an inverse!

Some keyboard commands can be called with an argument which specifies how many times the command is to be repeated. By default this argument is 1 but may be changed with the **C-U** key. To set the argument, input **C-U** followed by a non-negative number (the required number of repeats). At this stage the instruction can be aborted without specifying a command by pressing **C-G**. Otherwise the command is input and will be executed the required number of times. For example, **C-U 20 C-V** will scroll the document on by twenty pages. If it is not meaningful to repeat a command then the argument will be ignored and the command executed only once.

7.1 Moving the Cursor

The following commands move the cursor or the target selection.

C-F Move forward one character.

$\rightarrow$ Move forward one character.

C-B Move backward one character.

$\leftarrow$ Move backward one character.

M-F Move forward one word.

M-B Move backward one word.

C-N Move to the beginning of the next line.

$\downarrow$ Move to the beginning of the next line.

C-P Move to the end of the previous line.

$\uparrow$ Move to the end of the previous line.

C-E Move to the end of the current line.

End Move to the end of the current line.

C-A Move to the beginning of the current line.

Home Move to the beginning of the current line.

M-$<$ Move to the beginning of the document.

M-$>$ Move to the end of the document.

M-R Move to the centre line of the window.

7.2 Scrolling the Window

That portion of a file displayed in a window can be scrolled using the scrollbars (see section 4.3.1) or with the keyboard commands listed below. Note however, that the position of the cursor or the target selection remains unchanged by these actions even when no longer visible in the window.

C-L Redraws the window and repositions the document so that the cursor or target selection will be placed at the middle line of the window.

C-V Scrolls the document one page up.

M-V Scrolls the document one page down.

Since **C-V** and **M-V** do not alter the cursor position or target selection, **C-L** can be used to scroll back to it again.

7.3 Erasing and Retrieving Text

The commands to erase and retrieve text are listed below. Note that "deleted" or "killed" target terms can always be retrieved from the buffer. However if the target is empty (i.e. the black diamond edit cursor is visible) "deleted" items cannot be retrieved whereas "killed" items are stored in a system buffer and can be retrieved using the "yank" operation.

C-D Delete the target or the character to the right of the cursor.

Del Delete the target or the character to the right of the cursor.

C-H Delete the target or the character to the left of the cursor.

Backspace Delete the target or the character to the left of the cursor.

C-K Kill all the characters to the right of the cursor up to the next expression or newline character. If the cursor is at a newline character or immediately to the left of an expression or template, this will be killed.

M-D Kill the target or the word to the right of the cursor.

M-Del Kill the target or the word to the left of the cursor.

C-W Kill the target selection.

CM-W Append the text which is killed (with the command that follows this one) to the most recently killed text.

C-Y Retrieve the most recently killed text and insert it at the position of the cursor.

The killed text and expressions can be expanded as long as the kill commands are used. Any other action will end the killed text following which a new kill command will place the text in the buffer from where it can be restored later.

Transposing text

The following two commands can be used to interchange the order of two textual units.

C-T Transposes the character to the left with that to the right of the cursor and moves the cursor one position to the right. Many typing errors can be recovered using this command. Note that an expression is handled as if it were a single character and so is moved in its entirety

M-T Transposes the words to the left and right of the cursor and moves forward one word. If the cursor is within a word, that word and the next one will be used. An expression is handled in this case as a single character that is not part of a word.

These commands can not be used to transpose parts of an expression.

7.4 Find and Replace

Incremental search and query replace commands based on the corresponding Emacs functions are available for use in Math∫pad. These commands can be called with the following key combinations.

C-S Initiate the search process which is to be in a forward direction starting from the target or cursor position. Math∫pad asks in the console for the search string and will search for each symbol as it is being entered. Note that a template may be entered but it must be the one and only symbol in the search string. During the search the following keys may be pressed or actions performed:

- **a click with the mouse on a template** If the search string is empty the template will be used as the search string. It, however, may not be edited. Otherwise the click on the template will be ignored.
- **press a letter, symbol or return on the keyboard** Add the character to the search string and search for the next occurrence of the new string.
- **C-S** Search in a forward direction for the next occurrence of the search string. If the last search action failed Math∫pad will start searching from the beginning of the document. If the search string is empty the last search string will be used.
- **C-R** Search in a backward direction for the previous occurrence of the search string. If the last search action failed Math∫pad will start searching backward from the end of the document. If the search string is empty the last search string will be used.
- **Esc** End the incremental search and leave the selection at the current position.
- **Del** Return the selection to the previous position and if appropriate delete the end character from the search string.
- **Backspace** As for **Del**
- **C-G** Cancel the incremental search and move the selection back to its original position.

C-R Start a backward incremental search. The same commands as for the forward incremental search (**C-S**) are used.

Note that although both incremental searches closely resemble the corresponding Emacs function not all the features of the latter are available.

M-% Initiate a query-replace command. Caution, before proceeding with this command ensure that the cursor is where you require it.

The query replace will ask for two strings, the string to be sought and the replacement string. Both strings must be of the same type and there are two types possible. The first type is a sequence of characters or symbols input from the keyboard or a symbol palette and terminated with a **return**. The second type is a template input from a stencil palette. Thus a string consisting of a mixture of templates and other symbols is not allowed.

After the two strings have been entered Math∫pad will search for occurrences of the search string within the scope of the query-replace command. The scope is either the source selection if this is set and is visible in the window or it is the portion of the document stretching from the target or cursor up to the end of the document. If an occurrence of the search string is found the user is asked to respond with one of the following inputs:.

y, Spacebar Replace this occurrence of the first string with the second string.

n, Del Skip this occurrence.

! Replace all remaining occurrences of the first string with the second string without prompting and move the cursor to the last replacement.

q, Esc Stop the query replace leaving the cursor or target at its present position.

C-G Stop the query replace and move the cursor or target back to its original position. Replacements already carried out are not undone.

Note that the replacement of a template will also replace the information concerning the kind of template, spacing and precedence. Missing arguments for an operator are not added.

The incremental search and query replace can also be terminated by clicking in an edit window.

7.5 Miscellaneous

The following additional commands are available to make working with Math∫pad a little easier.

M-L Math∫pad asks for a number n in the console and then moves the cursor to the line which produces the nth line of LaTeX code.

C-. Remove the dots displayed around a text place holder or, alternatively, if they are not displayed, show them. These dots can be used to check that the correct selection is made. They are also used to be able to place the cursor after the last symbol in a text place holder.

CM-left, CM-right
Move the displayed expression, entered with the Insert Disp button from the console or the **F6** key, to the left or right by one tab distance. While all displayed expressions should start at the same distance from the left margin, it could be that some expressions need extra or less indentation.

Furthermore, some keys receive special treatment when they are used in expressions.

The **space** character can not be inserted into an expression or identifier and when this is tried, the editor will move the selection to the next empty place holder or text position, which ever comes first. If a text position is found, the space will be inserted at that position. Otherwise the space will not be inserted. For example, when an identifier is used in a text and the name is filled in, the name can be ended with a space character which will be inserted immediately after the identifier in the text. However, when the identifier is part of a larger expression with empty place holders after the identifier then the selection will move to the first empty place holder after the identifier.

The **C-return** key is similar except a return is entered in the text instead of a space. The **return** key can also be used to terminate an expression or identifier but in this case neither a return nor a space is entered into the text.

The "(" key will place parentheses around an expression or insert it in an identifier. The ")" key will insert a parenthesis in an identifier if the matching opening brace is in that identifier. Otherwise the smallest parenthesised expression is selected. The combination of those two keys makes it possible to insert an expression from left to right.

The **,** key will insert a comma after the identifier and insert a new identifier or expression after the comma. If it is not possible to do this, the comma will be inserted in the identifier.

In order to test the effect of a keyboard definition file the key combination **CM-** can be used. Subsequent keystrokes (whether or not combinations) are displayed in the console as they are received and interpreted by Math∫pad. The combination **C-A** terminates this feature and returns the system to its normal state.

7.6 Pasting

In order to facilitate the transportation of information between Math∫pad documents and other systems a primitive *pasting* mechanism is provided. The mechanism works by inserting text contained in the window port (a buffer managed by the window system) into a Math∫pad document or, vice-versa, placing information from a Math∫pad document into the window port. Text transported from a Math∫pad document to the window system via this port is first translated to LaTeX to ensure that it is accepted by other programs. For normal text, there will be no difference between the LaTeX code and the ascii representation.

The steps for pasting a part of a Math∫pad document into another window are as follows:

- Select the text (or expression) that should be converted to LaTeX with the near mouse button
- Press the **F10** key. The selection is now converted to LaTeX and will remain available until something is selected in another program. Using the selection in Math∫pad is still possible.
- Paste the selection in the other program, which is done by using the Paste key, mouse button or menu option. (Consult the documentation on your window manager to see how this is done.)

The steps to copy text from another program to Math∫pad are:

- Select the text in the other program.
- Make the text available for other programs. Some programs do that automatically, other programs have a key or a menu option for this. (Consult the documentation on your window manager to see how this is done.)

- Paste the selection in Math∫pad by using the **F3** or **F18** key. The latter key is labelled "Paste" on a Sun keyboard.

7.7 Templates

Templates can also be used to advantage in (non-mathematical) text. For example, templates are provided in the **latex2e** stencil for the selection of fonts and sizes. With this facility you can enter, for instance, tiny sans serif text or **large bold text**. Templates can also be used for section and chapter headings, for indexing, for footnotes, for quotations and so on. They are particularly useful as a mechanism to save you the chore of repeatedly entering the same information (with the attendant risk of error). For instance, you may wish to construct an "address" template containing versions for your postal address, your email address and your personal world-wide web URL. A single click is then sufficient to enter the information into a document. (If you exploit this possibility take care to get the information in the template definition exactly right. Computers do not *eliminate* error: the advantage of using them is that, unlike human beings, they do not *introduce* error. Their disadvantage is that any errors we introduce are faithfully reproduced, and potentially at a very high rate!)

To change the font (say) of text that has already been entered it suffices to set the target to the text to be printed in a different font and then click on the appropriate version in the TextStyle template in the **latex2e** stencil.

In general, if a template with one or more text place holders is used when a target selection has been set, the selected text will be inserted at the first available text place holder in the template.

Information within a text place holder can be edited like normal text and can contain the same items as normal text. Thus a text place holder may contain, for example, expressions and special symbols. To insert information in a text place holder, click on the place holder and and then proceed as usual.

Inside a tabbing environment LaTeX's automatic line breaking routines are disabled. Therefore when filling a text place holder in a tabbing environment the user should insert line breaks at the appropriate places. The proportionally-spaced font used in the screen version of the document is a help in this situation.

There is an automatic line wrap facility that can be turned on or off (turning off is not yet implemented). In addition a line wrap can be executed on the target via the console. See the section on console commands. There is no right-justify facility.

Chapter 8

Structure Editing

Structure editing, as opposed to text editing, can be enormously beneficial because it enables one to work with logical building blocks rather than with the encoding of those building blocks as sequences of characters. Structure editing is particularly beneficial to the creation of mathematical documents, and a very natural technique to use.

As an example of structure editing, suppose one states and proves a theorem in a mathematical document. It is often convenient to give the theorem a name in order to be able to refer to it later. Let us suppose the theorem is called the "Banchirikovsky theorem". (We have deliberately chosen a complicated name in order to emphasise our point.) When the name is first entered its "encoding" as a sequence of characters "B", "a", "n", "c", *etc.* is relevant, but thereafter it is not. It is the *name* that we use, not the sequence of characters. The name of a theorem can be given a logical role using a structure editor by making it a component in a "theorem" template. By so doing it is easy to copy the name, without any risk of introducing error. This is in contrast to the use of a text editor where it is the individual characters that one selects and copies, and where the risk of introducing error is present.

The structures in a Math∫pad document are embodied in the user-defined templates. In this chapter we assume that the templates are already available and we discuss the most basic edit operations that can be performed on structures (creation, copying and moving). See chapter 11 and chapter 12 for the creation of templates and chapter 9 for more advanced edit operations relating to the manipulation of mathematical expressions.

Text is not completely unstructured: it consists of "words" and lines (a "word" being a sequence of characters of maximal length delimited by spaces or line breaks). Facilities built into Math∫pad that exploit this structure are also discussed in this chapter.

8.1 Entering Templates

Expressions are entered into a document via the templates of a stencil. See section 5.2 for how to load a stencil.

The window in which a stencil is displayed is called a stencil *palette*. Templates from it can be entered into a document with a simple click of the mouse. (See the following section for the fine details.) The template is then entered at the position of the *target selection* (subject to a few restrictions).

Templates may be entered anywhere in running text. They may also be entered into place holders created by other templates. The only restriction on the latter is that templates may not be entered in identifier or variable place holders.

The type of the target selection determines how the template will be used. If the template cannot be used, the enter command is simply ignored (and no error message is given). Some examples of entering templates are given in table 8.1.

template	target selection	result
$\leq$		$\mathbf{E} \leq \mathbf{E}$
$\neg$	$\mathbf{E}$	$\neg\mathbf{E}$
^{2}	$a+b$	$(a+b)^2$
$\wedge$	$P \vee Q$	$(P \vee Q) \wedge \mathbf{E}$
$\wedge$	$\mathbf{O}$	$\wedge$
$\forall(\mathbf{V} : \mathbf{E} : \mathbf{E})$	$\neg R$	$\forall(\mathbf{V} : \neg R : \mathbf{E})$
$\underline{\mathbf{T}}$	text	<u>text</u>

Table 8.1: Effect of entering a template.

The first example illustrates the creation of expressions in running text. The target is empty, the chosen template is a binary operator, and the result is to enter two expression place holders separated by the chosen binary operator in the text at the position of the cursor.

The next two examples illustrate the introduction of parentheses. Note that parentheses are introduced around the expression $a+b$ in the third example whereas no parentheses are introduced in the second example. The rules governing their introduction are the normal mathematical conventions regarding precedence of mathematical operators, the precedence that has been given to specific operators being completely user-defined. In the case of the third example the precedence given to the squaring operator is 20 and that given to addition is 13. (This information can be obtained via the Version button on the console. See section 8.1.2 for further information.)

In the fourth example the target is an expression and the chosen template is that of a binary operator. The result is that the target becomes the left operand of the binary operator. (It is not possible to specify that the target becomes the right operand.)

The fifth example contrasts with the fourth example: if the target is an operator place holder rather than an expression or expression place holder then selecting an operator will insert just the operator, not the operator and (place holders for) its operands.

The template chosen in the sixth example is more complicated. It has two expression place holders and one variable place holder. The target is an expression. Math∫pad inserts the target in the first empty, type-correct place holder, in this case the first expression place holder. The order given to the place holders is specified in the definition of the template and is not necessarily left-to-right.

The last example shows that what goes for mathematical expressions also goes for text. It illustrates the technique mentioned in section 7.7 for specifying font or size information.

Naming place holders has no effect on the above rules. Suppose, for example, that one defines a programming language stencil with templates for statements, declarations, expressions and so on. Suppose also that all place holders are appropriately named so that the syntactic structure of the language is made clear to the user of the stencil. Then, it will nevertheless be the case that the user may enter a statement (say) where a declaration is required. The names of place holders are there to guide, not to discipline the user.

After using a template the target selection will be set to the leftmost empty place holder of the resulting expression. If there are no empty place holders, the edit cursor will be placed immediately after the expression. The target selection will therefore be at the most likely position at which the next expression or text will be entered.

To undo the insertion of a template is usually straightforward. If the target was empty before the insertion set it to the inserted expression and press the delete or backspace key. If the target was not empty before the insertion then set the source to the original target and the target to the inserted expression. Then execute the Copy command. (These instructions may be difficult to carry out if the template you selected involves complicated layout instructions, particularly where these conflict with layout information in the original target.)

8.1.1 Switching between Versions

Templates may have several *versions*. To switch between versions of a template set the target to the occurrence of the template in the document and with the far mouse button reveal the menu under the Version button on the console. The hand that appears indicates which version of the template is currently being used. Move the hand to point to the version required and release the mouse button. The target will alter accordingly.

This facility can be very useful, particularly for information which one only wishes to have displayed on the computer screen occasionally. For example, the footnote template has one version which displays the text of the footnote on the screen and one which doesn't. The former version must be used to enter or edit the text of the footnote but at other times either version may be used. (Of course both versions must produce the same output code.)

8.1.2 Help on Templates

To find out more, set the target selection to the template you are interested in. (You may sometimes have to move up or down in the tree structure to find the correct template.) Then select Version in the console with the far mouse button. Pin up the menu that appears and select Info. The information that appears contains the following items:

- The name of the template.
- The name of a help file.
- The operator type, its precedence and default spacing.
- The number of versions of the template.
- The number of times the template has been used.
- The names of loaded stencil files in which the template can be found.

The name of the template is the entry name under which the template appears in any stencil.

A help file associated with a template may or may not have been provided by the creator of that template. The contents of a help file is also the responsibility of the creator of the template. The intention, however, is that the help file explains the purpose of the template and gives examples of its use. A help file is just a normal Math∫pad document and so can be read by loading it via the Document entry below the Window button on the console.

8.2 Entering Place Holders

Occasionally it is necessary to create a place holder in a document (other than by entering a template). Examples are: the introduction of an identifier in running text (e.g. as in "let x be ..."), the insertion of a fresh mathematical expression that begins with a parenthesis, the inclusion of a displayed mathematical formula, and changing the kind of a place holder (for example, from expression to text). To do so the target is set appropriately —in all but the last example this means setting the cursor to the point in the text at which the place holder is to be inserted— and then one of the following methods is used to enter the place holder.

- Click on on the relevant menu item under the EditOp button in the console. The options are Insert Expr (insert expression), Insert Disp (insert display), Insert Id (insert identifier), Insert Op (insert operator), Insert Text (insert text) and Insert Var (insert variable).
- Type **CM-P** for a displayed expression. Otherwise type **CM-** < **char** > where < char > is the initial letter of the kind of place holder you require.

Expression and display place holders are entered most frequently. For this reason, they can be entered via function keys, **F5** for an expression and **F6** for a display.

8.3 Entering Expressions

Complete expressions can be entered by repeatedly entering the templates that make up the expression. Three techniques for building up a complicated expression are described below:

top-down An expression usually contains a number of templates and operators at different levels. The top-down technique first enters the main operators at the highest level. The empty place holders are then entered from left to right, again with the operators first. The cursor will automatically move to the first place holder after an operator is inserted, so you don't have to move the mouse between different windows.

bottom-up It is also possible to input a subexpression first and then add the operator from a higher level. In the bottom-up technique, the leftmost subexpression is entered first. The operators are entered later and the selected expression is used as the first argument of the operator.

left-to-right A third possibility is to enter the expression from left to right. This is the most natural method as it's the way we read.

Most commonly a combination of these techniques is used since in mathematical texts (sub)expressions are frequently repeated. The left-to-right technique is the most natural method as it's the way we read. It has the disadvantage, however, that it can involve much switching between the keyboard and the mouse. (This is much less so the more use is made of copying to create an expression.) It shares this disadvantage with the bottom-up technique. The top-down technique, on the other hand, has two phases. The first phase involves the use of the mouse to build up the operator structure, and the second phase uses the keyboard to fill in variables.

The difference between these three techniques is shown in table 8.2 with an example from Boolean algebra: entering the law $\neg a \wedge \neg b \equiv \neg(a \vee b)$.

In table 8.2, "*click on*" means click on a specific template in the stencil palette, "type" means press a sequence of keys, "*enter*" means press a sequence of keys followed by a **return** and "*select*" means select the expression with the target selection.

The table not only gives the number of steps that are required but also an indication of the physical work involved. The order in which the steps are performed shows how often the user has to switch between the mouse and the keyboard. A switch from "type" or "enter" to "click" or "select" means a switch from the keyboard to the mouse. A "select" followed by a "click" in a template will force the user to move the mouse to the edit window, click at the correct position and move it back to the stencil palette. If the number of switches and mouse movements between the edit window and stencil template are compared, it can be seen that there is not much difference between the top-down and left-to-right techniques (the former being slightly better). The bottom-up technique, however, clearly involves more work.

top-down	bottom-up	left-to-right
Click on $\equiv$	Insert Expr.	Click on $\neg$
Click on $\wedge$	Type `a`	Type `a`
Click on $\neg$	Click on $\neg$	Click on $\wedge$
Enter `a`	Select $\neg a$	Click on $\neg$
Click on $\neg$	Click on $\wedge$	Type `b`
Enter `b`	Type `b`	Click on $\equiv$
Click on $\neg$	Click on $\neg$	Click on $\neg$
Click on $\vee$	Select $\neg a \wedge \neg b$	Type `(a`
Enter `a`	Click on $\equiv$	Click on $\vee$
Enter `b`	Type `a`	Enter `b`
	Click on $\vee$	
	Type `b`	
	Select $a \vee b$	
	Click on $\neg$	

Table 8.2: Techniques for entering expressions.

Each technique uses some of the following features to make the editing easier.

- Pressing the **return** or **space bar** will move the target selection to the next empty place holder in the current expression. The top-down technique uses this each time an identifier is inserted.
- A selected expression will be used as the first argument of a template or operator. An existing expression can therefore be used as a subexpression. This is used in the bottom-up technique.
- If an operator template is selected after entering an identifier (and before taking any other action such as entering a space or return), the left operand of the operator is taken to be the subexpression containing only operators of a higher precedence that is completely to the left of the cursor. Parentheses can be used to ensure that a particular expression is used as the argument.
- The left parenthesis from the keyboard can be used to put parentheses around an empty place holder. The right parenthesis is used to select the expression within the parentheses. This is used in the left-to-right technique in order to enter $\neg(a \vee b)$ instead of $\neg a \vee b$.

- It is possible to insert a sequence of operators such as $a \wedge b \wedge c$ by selecting one part of the available expression and using the same operator again. This can be used in all techniques. In the top-down technique, you just click twice on $\wedge$. In the bottom-up technique, select $a \wedge b$ or b and click on $\wedge$. In the left-to-right technique, click on $\wedge$ after typing b

Remember that it is always possible to use a mixture of the techniques and features depending on the situation in hand.

8.4 Continued Expressions

Continued expressions are expressions like $a{+}b{+}c$ and $0 \leq i < N$ consisting of a sequence of infix operators and their arguments. Such expressions occur frequently in mathematics. Accordingly, their creation and manipulation causes no particular problems. The expression $a{+}b{+}c$ is entered by simply selecting the $+$ template twice (and filling in the variables a, b, and c). Entering $0 \leq i < N$ is equally simple provided only that the operators $\leq$ and $<$ are versions of the same template. A continued expression can also be created with (infix) operators from different templates but in that case the ungroup operator described in the next chapter has to be applied.

8.5 Selecting

Both text and expressions can be selected with the mouse. The selection of text will be the same as in many other window applications. By clicking, the cursor will be set and the previous selection will be undone. By dragging the selection can be set to the text between the position where the mouse button is pressed and where it is released. The mouse button indicates which selection is to be made — (in the default situation) near button for the target, middle button for the source, and far button for the argument selection — and all are made in the same way.

8.5.1 Text

A double click can be used to select other units and can be used after a click or drag. The possible units are characters, words, lines and a complete text. The units are used in the following way:

- A normal click or drag will select characters.
- A double click will select the word and with a double drag complete words are selected. A word consists of a sequence of letters and digits. Other characters and symbols are still selected one by one.
- A triple click will select a complete line, defined as the shortest sequence beginning and ending with a return. Templates may also contain returns and a selected line may therefore contain multiple lines on screen. With a triple drag, multiple lines can be selected.
- A quadruple click will select the complete text. This text does not have to be the complete document. It is possible to select the text from a text place holder with such a click.
- Clicking five times will select characters again.

As described earlier, the double click/drag is a click/drag that immediately follows a click or drag. The triple and quadruple click/drag are defined the same way. The only two positions used by a multiple click are the position where the first click/drag starts and the position where the last click/drag ends. These two positions determine what is selected.

8.5.2 Expressions

When an expression is selected, the selection will be the smallest expression that contains both positions as described above. If these positions are equal the smallest non-operator will be selected. This means that clicking on a symbol of a template will select that template and clicking on an operator will select the operator and its arguments.

Continued expressions like $a + b + c + d$ have continued expressions as subexpressions. (In the case of $a + b + c + d$ these are $a + b$, $a + b + c$, $b + c$ *etc.*) These may be selected by dragging from the first to the last operand. Sections of continued expressions (subsequences beginning or ending on an operator) can also be selected by commencing the selection appropriately. The selection will be shown during the drag. Some (but not all) selections from the sequence $P \vee Q \vee R$ are shown in table 8.3.

A double click can be used to select the complete expression or to select the expression that contains the selected expression as a subexpression. For example, parts of the expression $0 \leq i < N \vee j > 1$ can be selected as shown in table 8.4.

not sectioned	sectioned
P	$P \vee$
$P \vee Q$	$P \vee Q \vee$
$Q \vee R$	$\vee Q \vee R$
$P \vee Q \vee R$	$\vee Q \vee$

Table 8.3: Sectioned operators.

Action	On symbol	Resulting selection
Click	$\leq$	$0 \leq i$
Double click	$\leq$	$0 \leq i < N$
Triple click	$\leq$	$0 \leq i < N \vee j > 1$
Click	$\vee$	$0 \leq i < N \vee j > 1$
Click	N	N
Double click	N	$0 \leq i < N$
Click	$<$	$i < N$
Drag	$\leq$ and $<$	$\leq i <$
Drag	N and j	$0 \leq i < N \vee j > 1$
Drag	j and $\vee$	$\vee j > 1$

Table 8.4: Multiple clicks and drags.

A double click can also be used to select templates that are not directly visible on screen, whereas a normal click will only select the argument of the template, not the template itself. After a double click on the argument the template is selected. This often happens with templates that only change the font style or the size of the text.

8.5.3 Large Selections

Large selections that don't fit on one screen can be made using the Join function available via the Selection button on the console. This function expands one selection to contain another selection. In a submenu, you can choose which selection you want to expand: the target selection, source selection or argument selection.

As an example, to copy a large part of text from one document to another document, follow these steps:

- Set the source selection at one end of the text.

- Set the argument selection at the other end of the text.
- Select the item S+A $\rightarrow$ S from the Join menu. The source selection will be extended to contain the argument selection.
- Set the cursor at the position where the text should be inserted.
- Click on the Copy button or use the **F7** keyboard shortcut.

8.6 Copy and Swap

The Copy and Swap functions are used to copy, swap or move both text and expressions. The functions are available through the options Copy and Swap in the EditOp menu and the keyboard shortcuts **F7** and **F8**. To *copy* a piece of text to a particular position, follow these steps:

- Select the text to be copied with the source selection.
- Set the cursor at the target position.
- Select the Copy option or use the **F7** shortcut.

If the target selection is not empty, it will be moved to the buffer first.

To *swap* two pieces of text, the steps are almost the same:

- Select one piece with the target selection.
- Select the other piece with the source selection.
- Select the Swap option or use the **F8** shortcut.

If the target selection is empty, the swap function can be used to *move* text from one position to another. The cursor determines the target position of the move in that situation.

Other kinds of structure (expressions, operators *etc.*) can also be copied or swapped. Swapping may even involve incompatible kinds of structure, with some restrictions. Table 8.5 shows where the restrictions occur.

Target	Source	Copy?	Swap?
Text	Text	Yes	Yes
Text	Not a text	Yes	Sometimes
Expression	Expression	Yes	Yes
Expression	Operator, identifier	Yes	No
Expression	Text	Sometimes	Sometimes
Operator	Operator	Yes	Yes
Operator	Not an operator	No	No
Identifier	Identifier	Yes	Yes
Identifier	Not an identifier	No	No

Table 8.5: Restrictions on Copy and Swap.

The “Sometimes” entries are explained by the fact that in-line expressions (and identifiers *etc.*) may occur in text, but the movement of text may not result in its occupying a non-text placeholder within an expression.

The move function has the same restrictions as the swap function.

The copy function also works if the target selection is part of the source selection or vice versa. The result will be what you would expect: the target selection is replaced by a copy of the source selection. Some examples are given in table 8.6.

Target	Source	Result
`{\it` italic `\/}`	italic	italic
$a \vee (b \wedge c)$	$b \wedge c$	$b \wedge c$
$R \vee S$	$P \Rightarrow R \vee S$	$P \Rightarrow P \Rightarrow R \vee S$

Table 8.6: Copying when source or target is a subexpression.

8.7 Undoing Changes

Math∫pad has no functions to undo changes. As a safeguard against error, deleted elements are stored in the *buffer*. The contents of this buffer is displayed in the *buffer window*, as shown in figure 8.1. This window is used like a normal document window; deleted text and expressions can be restored by copying them back to a document window.

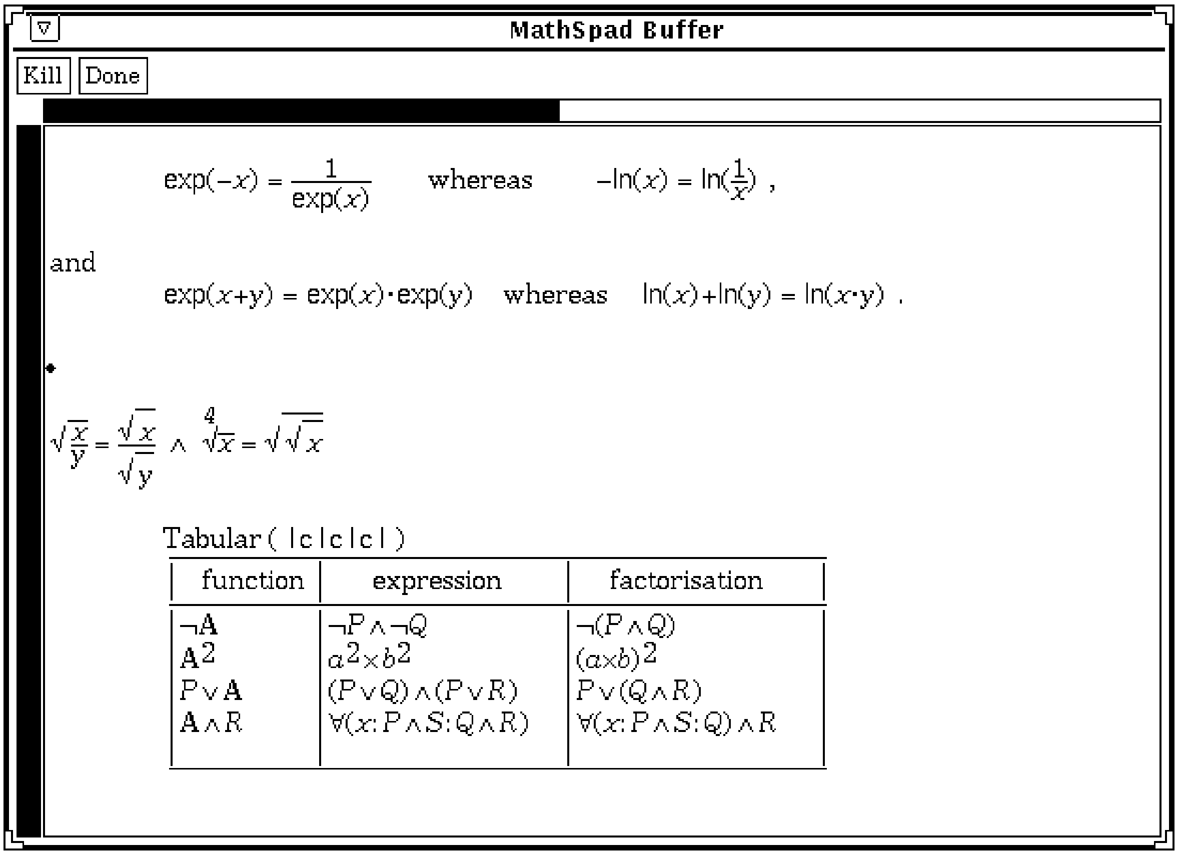


Figure 8.1: The Buffer Window

New deletions are added at the beginning of the buffer. Opening the buffer window will show you what has been deleted. The following functions may add something to the buffer:

- Copy, if the destination is not empty.
- Delete or backspace, if the target is not empty.
- Using a template, if the selection can't be used in that template.
- A new kill sequence, if the kill buffer is not empty.
- Replace, if the argument can't be used in the replacement.

So, during a session, the contents of the buffer window will grow.

If you delete something in the buffer, it will not be stored again and an undo is not possible. This can be used to clean up the buffer or clear it completely. However, if you kill something in the buffer, it can be recovered via the kill buffer. You can clear the buffer also by clicking on the Kill button.

8.8 Information on Templates

It is often the case that you want help with the use of templates, even with templates that you yourself have created. Math∫pad provides several mechanisms for doing so which we discuss in this section.

8.8.1 The Document Stencil

The most comprehensive information about a template is provided by its definition. See chapter 12 for how to (re)define a template. By going through the motions of redefining a template, but not making any change whatsoever, you can find out full details of any template.

A stencil, called the *document stencil*, containing all the templates used in a document can be generated automatically as follows: Click on the Stencil menu item under Window on the console, and then click on Load in the stencil window that appears. The file selector will appear on your screen, the message "Load stencil:" at its head indicating that its function is to load a stencil. The "Mask" entry will accordingly be set to

"`.mps`". Edit the mask so that it reads "`.mpd`", and then seek the document that you are interested in. (See section 5.4 for details of how to do this.) Once the document name has been found and entered as the "Selection" click on OK. A stencil will appear on your screen; the name of this stencil is the name of the document you selected. (Thus if you selected the document "`mydocument.mpd`" the header of the stencil will read "mydocument.mpd Stencil".

The principal use of the document stencil is as a summary of all the templates that have been used in a document. It can also be used to recover the definition of templates that have been lost (because the stencil in which they were first defined has been lost or destroyed or because the definition has been inadvertently overwritten). Document stencils should not normally be saved since there is little point in doing so.

8.8.2 Info and the Help File

Summary and help information about a template can be obtained via an occurrence of the template in a document or in a stencil. The information that is obtained is identical in both cases but the procedure to obtain the information differs.

In the case of an occurrence of a template in a document the procedure is as follows. First set the target selection to the template you are interested in. Then select Version in the console with the far mouse button. Pin up the menu that appears and select Info.

In the case of an occurrence of a template in a stencil a similar menu can be obtained by selecting the template with the far mouse button. Pin the menu up and select Info.

(Note that the headers of the two menus differ. The menu obtained via Version in the console is headed "Version" and the menu obtained via a stencil is headed "Template". This has nothing to do with the information that is available.)

The information that appears contains the following items:

- The name of the template.
- The name of a help file.
- The type (mathematical operator or not) and, if a mathematical operator, its precedence and default spacing.
- The number of versions of the template.

- The number of times the template has been used.
- The names of loaded stencils in which the template can be found.

The name of the template is the entry name under which the template appears in any stencil.

A help file associated with a template may or may not have been provided by the creator of that template. The contents of a help file is also the responsibility of the creator of the template. The intention, however, is that the help file explains the purpose of the template and gives examples of its use. A help file is just a normal Math∫pad document and so can be read by loading it via the Document entry below the Window button on the console.

Chapter 9

Algebraic Manipulations

Math∫pad is oriented to the creation of mathematical documents. Syntactic, algebraic manipulation of mathematical expressions is therefore a key component.

A feature of algebraic calculations is that the same (sub)expressions are used repeatedly in a calculation with relatively small changes in between. The copy and swap functions, as described in section 8.6, are thus the mathematician's favourite tools. In addition to copy and swap, the following elementary operations form the basis of a large number of algebraic rules:

- Reversal (Commutation) of subexpressions.
- Identifier renaming.
- Applying functions.
- Distribution of a function over the terms of an expression.
- Factorisation of a function out of the terms of an expression.

These structure editing operations are described in this chapter. Additional structure editing operations described here are the addition and/or removal of parentheses, and grouping or ungrouping of subexpressions. All are to be found under the Structure button on the console.

Specific algebraic laws can be applied using the find-and-replace facility described in chapter 10.

9.1 Grouping and Parentheses

In Mathƒpad *continued expressions* (operator-argument sequences such as $a+b+c$) are made by selecting the same template repeatedly. For example, $a+b+c$ is entered by selecting the $+$ template twice. Continued expressions in which the operator is not constant but the operators are versions of the same template, like the sequence $0 \leq i < N$, are created in the same way: (assuming the operators "$\leq$" and "$<$"are both versions of an ordering relation template) insert an expression, type 0, click on the version "$\leq$" of the ordering relation template, type i, click on the version "$<$" of the ordering relation template and, finally, type N.

The *ungroup* operation is used when it is required to create a continued expression in which the operators are not versions of the same template. For example, the expression $0 \;\leq\; i \;=\; N$ (in which the operators "$\leq$" and "$=$" are versions of different templates but have the same precedence) can be constructed as follows:

- Construct $0 \leq (i = N)$ in the usual way.
- Select $i = N$ by clicking on "$=$".
- Click on the Ungroup button. The parentheses will now disappear and the structure of the expression will be changed.

There is a restriction on the use of the ungroup button: all the operators in a continued expression must have the same precedence.

The difference in structure between a continued expression like $0 \leq i = N$ and a non-continued expression like $a + b{\cdot}c$ is evident from the selections that can be made. In the former case there are many subexpressions that can be selected —$0 \leq i$, $i = N$, $0 \leq$, $0 \leq i =$, $\leq i =$, etc.—, in the latter case there are fewer selections possible — it is not possible to select $a+b$ as a subexpression, for example— . This affects in turn the effect of a reverse, distribute or factorise operation, as well as the insertion of parentheses. The ability to work directly with continued expressions is a significant contribution to Mathƒpad's handiness when performing algebraic calculations.

The reverse action is also possible: a subsequence of a continued expression can be made a proper subexpression of the expression. This is done with the Group button. The effect of Group is normally visible by the spacing around the operators. It does not cause parentheses to be inserted.

The Insert/Rem () button has an action that is similar to but not the same as Group or Ungroup. The insertion of parentheses is commonly used when the writer wants to be more explicit about the structure of an expression. For instance, one may want to parenthesise the expression $a = b \wedge c$ to create $(a{=}b){\wedge}c$ even though the precedence of the operators as defined in the stencil does not require it. Removal of parentheses undoes such an action.

When parentheses are to be inserted around a subsequence of a continued expression the subsequence will first be grouped making it a subexpression of the entire expression. Subsequent removal of the parentheses is effected by Ungroup and not by Insert/Rem ().

To illustrate the differences consider the expression $a = b = c$. Applying Insert/Rem () to target $b = c$ will result in $a = (b = c)$. The parentheses are removed by applying Ungroup to the target $b = c$. Applying Group to the same target in the original expression creates $a \;=\; b = c$. (Note that the spacing around the first equals sign is bigger than around the second.) Selection of $b = c$ again and applying Ungroup returns the expression to its original form.

The choice between insertion and removal of parentheses is dependent on whether or not the target expression is already parenthesised. If it is, an attempt is made to remove them; if not, parentheses are inserted. Removal of parentheses will fail if the precedence rules do not allow it, as in writing $a{\times}b{+}c$ when $a{\times}(b{+}c)$ is meant. You can insist on removal by using the **F4** key.

9.2 Using Versions

Math∫pad templates typically come in several versions. This simple idea finds all sorts of uses: changing the layout of expressions, hiding information from a reader that is nevertheless needed by the writer, organising the contents of a stencil etc. The use of versions of a template has already been discussed in the context of continued expressions. In this section we discuss some other uses of versions.

Math∫pad does no automatic layout formatting. Formatting large expressions is notoriously difficult and there are lots of different conventions. So, instead of providing a complicated automatic formatting feature with lots of options, an easy to use manual formatting feature is available, which works for every style of breaking up large expressions.

The possibility to use templates with different versions enables you to divide an expression over multiple lines by changing to another version. Every layout style can be incorporated by defining the correct templates and versions. Define one version for the single line operator and another version for the multiple line operator and the layout of the operator can be changed.

Changing between versions is very simple. Just set the *operator* or *template* to the target selection. Then clicking on the Version button of the console will cycle through the different versions of the template.

Version
Bold
Italic
Bold italic
Slanted
Sans Serif
Typewriter
Emphasis

Table 9.1: Example of a version menu.

Alternatively, the required version can be chosen directly. By clicking with the far mouse button on the Version button, a pin-up menu like the one shown in table 9.1 will appear with the available versions. The current version is the one pointed to by the mouse; by moving the mouse another version can be selected.

The different versions can also be used to add or remove an argument to an operator or to change the layout in another way. For example, a font-changing template can be used to set a portion of text in another font. After selecting the template, the version menu can be used to select another font (from **bold** to typewriter, for example) or to change between, say, $\sum_{i=0}^{N} i$ and $\displaystyle\sum_{i=0}^{N} i$.

Note that different versions of a template may have different place holders. This is useful to hide information. For example, you may wish to produce two versions of a document, one including complete proofs of every theorem, the other containing only selected proofs. By including two versions of the "theorem" template, one which shows and one which doesn't show the proof, it is possible to quickly and accurately produce the two versions of your document. In switching to the version omitting the proof the contents of the "proof" place holder are not lost: switching back will restore the information in its entirety.

The only requirement on versions of a template is that the first version should always contain all place holders. Other versions may omit place holders. For further details see chapter 12.

9.3 Fine-Tuning the Spacing

Mathspad uses the spacing around operators to display the structure of mathematical expressions. To this end so-called *elastic space* is placed between an operator and its arguments. This space grows automatically as the complexity of the operands increases. The expression below is an illustration.

$$a + b{\times}b \;=\; c \;\;\wedge\;\; d/e \leq f \quad\equiv\quad b + a{\times}b \;=\; c \;\;\Rightarrow\;\; d \leq e$$

The principal operator in the expression is "$\equiv$"; accordingly this operator has the most space around it. The other two logical operators, "$\wedge$" and "$\Rightarrow$" have the next lowest precedence; the amount of space around them is a little less than around the "$\equiv$" operator and a little more than around the "$=$" and "$\leq$" operators. The operators that have the least space around them are the arithmetic operators, which have the highest precedence. Unlike in TEX, the spacing around operators is computed dynamically rather than being constant. Compare the spacing around the "$\equiv$" operator in the above expression to that around the same operator in the expression below.

$$a \equiv b$$

An algorithm to compute the desirable amount of space that works in all situations is impossible to construct —what is acceptable to the human eye depends on a lot of factors including the size and shape of the operator symbols— and so a facility is provided to change the spacing in situations where the calculated spacing is unsatisfactory. The following keys can be used for this: These changes are relative to the space calculated

Key	Action
C-$-$	Decrease the space.
C-$+$	Increase the space.
C-$=$, **C-0**	Reset the space to the default.

Table 9.2: Fine-tuning spacing.

automatically. This fine tuning is usually not needed until you have finished writing your document and are preparing the final output.

9.4 Reverse

Often, the arguments of a continued expression must be placed in a different order. This can be done with the functions copy and swap, but the Reverse button or **F9** shortcut can be used to reverse the order of the expressions and operators of a continued expression in one go. The longer the expression the more useful this function can be. The expression

$$0 \leq i < j < N$$

can be transformed into

$$N < j < i \leq 0$$

by using three swap operations and six selections. Using the reverse button it can be done with one operation and one selection.

As evident in this example, both the operators and the arguments appear in reverse order. This can be used to reverse a calculation or derivation, the comments remaining in the correct position. The example also shows that the operators are not replaced. The logical reversal would be $N > j > i \geq 0$. The database needed to do this is not available.

When the target has no subexpressions it will be swapped with the expression immediately to its left (right if left is not available). By default, Math∫pad uses the selected expression as first argument of the operator. The selection will be the other argument of the operator and the reverse function (**F9**) can be used to switch the two arguments.

9.5 Renaming Identifiers

Identifier renaming can be used to change all occurrences of an identifier in an expression at once. The notions of variable place holder and identifier place holder give meaning to the rename operation. Specifically, all identifiers listed in a variable place holder of a template are bound to that place holder. For example, defining a quantifier template to have the form $\mathbf{O}(\mathbf{V} : \mathbf{E} : \mathbf{E})$ (where **O** is an operator place holder, **V** is a variable place holder and **E** is an expression place holder), will cause all occurrences of the identifiers i and j to be bound to the variable place holder in the expression $\sum(i,j : 0 \leq i < j < N : i^2 + j^2)$. Renaming i to k results in the expression $\sum(k,j : 0 \leq k < j < N : k^2 + j^2)$.

To use the rename function, follow these steps:

- Select the identifier that must be renamed with the target selection.
- Select the identifier that contains the new name with the source selection. (If the identifier doesn't occur in your text you may wish to insert it in, say, the buffer first.)
- Select the scope of the rename function with the argument selection.
- Click on the **Rename** button.

The identifiers within the scope are now renamed.

The renaming function will check if an identifier is bound to a variable place holder in a subexpression. If so the template to which the variable place holder belongs will be skipped by the rename operation. The renaming function does not check for so-called "capture" of free variables. For instance, renaming i to j in the expression $\sum(i:\ 0 \leq i < j < N:\ i^2 + j^2)$ will capture j. The result is meaningful but probably not what is intended.

It is possible to use the rename function without a scope. If the identifier selected by the target selection is part of a variable list and the scope is empty, then all the identifiers bound to that identifier are renamed. If two normal identifiers are selected, the rename will perform a normal copy.

In the following examples, the identifier i is replaced by k using the rename function.

Scope	Renaming i with k.
$\forall(i:\ i \in \mathbb{Z}:\ i^2 + i > k)$	$\forall(k:\ k \in \mathbb{Z}:\ k^2 + k > k)$
$N > i \vee \exists(i:\ 0 \leq i < N:\ P.i)$	$N > k \vee \exists(i:\ 0 \leq i < N:\ P.i)$
For all positive i, $i^3 > 0$	For all positive k, $k^3 > 0$

Table 9.3: Effect of renaming.

In the first example, every occurrence of i is replaced by a k. The i is bound but the scope of the binding expression is equal to the scope of the rename. Rename can be called in two different ways: by setting the scope and selecting an arbitrary occurrence of i and k or by selecting the variable i, an arbitrary k and no scope. After the rename the last k is also bound because an identifier is bound on base of name equivalence.

In the second example, only the first i is not bound and the rename function will therefore only change the first i. (In a well-written document, this kind of expression is usually avoided.)

The last example looks like a normal rename but the function can be called in two ways. The first i has been inserted with the Insert Var. command and is the only element of a variable list. All occurrences of the identifier i throughout the document not bound to another variable place holder are thus bound to this variable. Renaming i will change all the i's used in the text.

9.6 Selecting Functions

In calculations functions are often used. In Math∫pad, functions are used for application, distribution and factorisation. A function is selected as follows:

- Select the *body* of the function with the source selection.
- Select the *argument* of the function with the argument selection.

The argument must be a subexpression of the body and the argument selection is therefore part of the source selection. The body of a function can be any expression and the argument does not need to be a place holder. So, the following expressions are all functions, where the argument is marked by "**A**":

$$\neg\mathbf{A}$$

$$\mathbf{A}^2 + 5$$

$$\sum_{i=0}^{N} \mathbf{A}$$

$$\forall(x: \ x \in \mathbb{R}: \ \mathbf{A})$$

The function can only have one argument. In some situations that is not enough. The copy and swap operations can be used in that situation or the find & replace feature can be used.

9.7 Applying a Function

A selected function can be applied to an existing expression. The expression is replaced by the function and the old expression is used as the argument of the function. This operation is a shortcut for a number of copy operations.

After selecting the function with the source and argument selection, the function is applied to the target selection by selecting the Apply option in the Structure menu. An example (the argument is given as $\mathbf{A}$) is shown in table 9.4:

function	expression	application
$\neg\mathbf{A}$	$a \wedge b \wedge c$	$\neg(a \wedge b \wedge c)$
$\mathbf{A}\star$	$s{+}t{+}$	$(s{+}t{+})\star$
$\mathbf{A} \vee R$	$P \wedge Q$	$(P \wedge Q) \vee R$
$R \wedge \mathbf{A}$	$\forall(x:\ P:\ Q)$	$R \wedge \forall(x:\ P:\ Q)$
$\forall(i:\ i{\in}\mathbb{Z}:\ \mathbf{A})$	$i^2 \geq 0$	$\forall(i:\ i{\in}\mathbb{Z}:\ i^2 \geq 0)$

Table 9.4: Effect of "applying" a function.

The selected function will not be removed and can be a part of the selected expression. The expression

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

can be changed to

$$a + b{\times}(a + b{\times}c)$$

by selecting $a + b{\times}\mathbf{A}$ as function and $a + b{\times}c$ as target.

9.8 Distribution

Distribution is one of the operations that is commonly used during the construction of a large expression with multiple repeating terms. For that reason a primitive distribution operation is available via the Distribute option in the Structure menu. To be able to distribute you must select a function and an operator sequence or template. When the distribution is performed the expressions in the root of the selected operator sequence or template are replaced by the selected function applied to these expressions. See table 9.5 for some examples.

The original function will be removed if you do not use the argument selection, but let the target selection be both "victim" and argument. Setting the target to $a{\times}(b{+}c)$ and source to the subexpression $b{+}c$ and with no argument selection, then the result of the distribution will be

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

just as one would expect.

function	expression	distribution
$\neg\mathbf{A}$	$a \wedge b \wedge c$	$\neg a \wedge \neg b \wedge \neg c$
$\mathbf{A}\star$	$s{+}t{+}$	$s\star + t\star +$
$\mathbf{A} \vee R$	$P \wedge Q$	$(P \vee R) \wedge (Q \vee R)$
$R \wedge \mathbf{A}$	$\forall(x:\ P:\ Q)$	$\forall(x:\ R \wedge P:\ R \wedge Q)$
$\forall(i:\ i{\in}\mathbb{Z}:\ \mathbf{A})$	$i^2 \geq 0$	$\forall(i:\ i{\in}\mathbb{Z}:\ i^2) \geq \forall(i:\ i{\in}\mathbb{Z}:\ 0)$

Table 9.5: Examples of function distribution.

When the selected operator sequence contains only one expression the result will be the function applied to the expression.

9.9 Factorise

Factorisation is the inverse of distribution and will factor out a common subterm from an operator sequence or template. The operation is available via the Factorise option in the Structure menu. To be able to factorise a function and operator sequence are needed. The function must be selected with the source and argument selection while the operator sequence is selected with the target selection. After clicking on the button the factorisation will be performed. Some examples are shown in table 9.6.

function	expression	factorisation
$\neg\mathbf{A}$	$\neg P \wedge \neg Q$	$\neg(P \wedge Q)$
$\mathbf{A}^2$	$a^2 \times b^2$	$(a{\times}b)^2$
$P \vee \mathbf{A}$	$(P \vee Q) \wedge (P \vee R)$	$P \vee (Q \wedge R)$
$\mathbf{A} \wedge R$	$\forall(x:\ P \wedge S:\ Q \wedge R)$	$\forall(x:\ P \wedge S:\ Q) \wedge R$

Table 9.6: Examples of factorisation.

If the function is not found in a subexpression,the subexpression will not change. In the last example $\wedge R$ is not found in the first expression (the expression $P \wedge S$) and so the expression does not change. In the second expression it is found so it will be removed and applied to the selected template. Distribution is therefore not the inverse of factorisation.

Chapter 10

Find and Replace

Every editor has a possibility to search for text and, usually, a replace function is also available. In Math∫pad incremental search and query replace are available for both text and templates. These functions are based on those provided by Emacs. A special window, the find & replace window, is provided for this purpose.

10.1 The Find & Replace Window

The find & replace window, as shown in figure 10.1, looks like a normal edit window, except that it has two equal-sized windowpanes. Each windowpane contains a separate document which is edited like a normal document. The upper pane contains the expression or text to search for, the lower one contains the replacement. We call these the find and replacement expressions, respectively. (Don't be confused by our use of the word "expression". Both "expressions" may well be just simple text.)

At the top of the window, there are three buttons, Find, Stack and Done. Under the Find button is a menu containing all the possible search and/or replacement commands. Those available are:

Find Search for the expression in the upper pane starting from the position of the target selection or cursor.

Replace Replace the found expression with the expression in the lower pane.

Replace&Find Perform a Replace and then a Find.

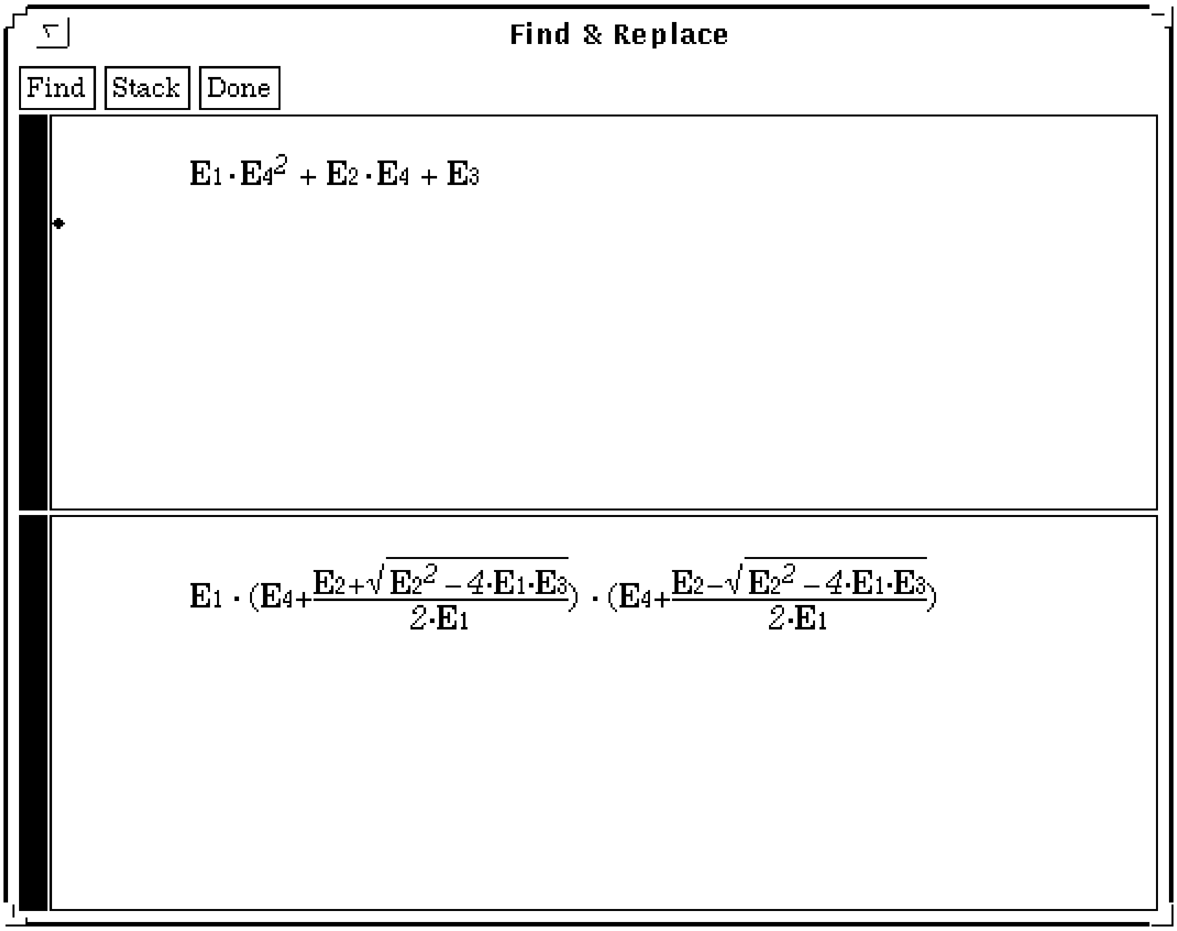


Figure 10.1: The Find & Replace Window.

Replace All Replace all occurrences of the upper expression from the current position of the target onwards with the lower expression.

Find & replace actions may be parameterised by including place holders in the entered expressions. (See fig. 10.1.) Such place holders are indexed, the indices being used in the obvious way to relate subexpressions in the find and replacement expressions.

The following checks are carried out before each find & replace action.

- Two place holders with the same index must have the same type.
- Each indexed place holder in the replace expression occurs at least once in the find expression.

Both the replacement and the find expression can contain multiple copies of the same place holder. So, for example, you can search for $\mathbf{E}_1 \wedge \mathbf{E}_2 \wedge \mathbf{E}_1$ and replace it with $\mathbf{E}_1 \wedge \mathbf{E}_1 \wedge \mathbf{E}_2$ or $\mathbf{E}_1 \wedge \mathbf{E}_2$.

A non-indexed place holder in the find expression matches an empty place holder in the document being searched. A non-indexed place holder in the replace expression enters an empty place holder into the document when the replace action is executed.

The rules for replacements are the same as the rules for copying: the type of the replacement expression must be the same as or convertible to the type of the find expression. Thus, replacing text by an expression is possible, but the reverse isn't. In some situations the conversion is not correct, for example, replacing $P \vee Q$ with $\forall(\mathbf{V} : P : P \vee Q)$ in the sequence $P \vee Q \vee R$. This problem will be resolved in a future release.

The target after a Replace is the replacement expression. In other words, a Replace does not have any side effect on the target selection such as moving it forward or backward in the document. This allows the possibility of a "recursive" replacement. For example, suppose the expression in the upper pane is the text "a" and the replacement in the lower pane is "bab". Then, a Find followed repeatedly by Replace and Find (or equivalently Replace&Find) will replace "a" successively by "bab", "bbabb", etc. This can be useful when editing mathematical expressions but, more importantly, it means that Replace All is *not* the same as repeatedly executing Find followed by Replace. (It would otherwise be very easy to send the system into an endless loop. The speed of conventional computers is also such that, even were we to provide a "cancel" button, the system would have exhausted all the available memory and crashed before the user had a chance to intervene!) To guarantee termination at all times, Replace All moves the target forward after each Replace. “Replace all” thus means “replace all currently existing occurrences” and does not mean that there will be no remaining occurrences after completion of the action.

The Stack button provides a primitive facility to record find & replace actions. These include saving find & replace actions in a file, loading previously saved actions, and stacking and retrieving actions within a session. The options listed are detailed below.

Load... Load a set of find & replace actions from a previously saved file.

Save... Save a set of find & replace actions in a file.

Prev Replace the current find & replace action by the one previous to the current action on the stack.

Next Replace the current find & replace action by the next action on the stack. The current contents of the find & replace window is simultaneously recorded in the stack, an empty entry causing the corresponding stack entry to be removed. The find & replace window will be cleared if there is no next action.

New Clear the find & replace window without recording the current action.

Clear Clear the contents of the find & replace stack.

The Load... and Save... open the file selector with the mask set to `.mpr` this being the standard extension for find & replace action files. No warning is given if stacked actions have not been saved when exiting from the system.

Repeated use of Next will cause the stack to be built up. Entries in the stack can be removed by returning to them with Prev, clearing the entry with New and then moving forward once more with Next.

A click on the Done button will close the find & replace window.

Note that the find & replace functions may be used with both text and expressions.

10.2 Special Editing Functions

The place holders in the find & replace window may or may not have indices. (A place holder with no index will match only with an empty place holder.) Indices are used to make a connection between different arguments and are therefore very important.

Math∫pad will automatically number the indices. Each new place holder will get the first unused index. The indices will change while you are editing the expression, so change the indices only after you have finished the expression. You can remove an index with the backspace or delete keys.

The place holders are used like normal expressions. All the usual functions work also on place holders with indices. You can copy indexed place holders from one position to another, swap two (indexed) place holders, reverse a sequence of place holders or distribute an expression containing an indexed place holder over any sequence.

If you wish to add an index to a non-indexed place holder, you can use the **C-I** key in combination with the universal argument. Thus to add index 4 to a place holder, select the place holder and type **C-U 4 C-I**. You can also replace an existing index with a new one with the same function.

10.3 Some Examples

The find & replace window can be used for finding text, templates, identifiers, operators or parameterised expressions of a certain shape. In combination with the replace function, spelling errors can be corrected and mathematical rules can be applied. The find & replace window is thus a powerful, general-purpose utility. Just how versatile it is is illustrated by a few representative examples.

To search for normal text just enter the text in the upper part of the window, place the cursor at the start position and click on the Find button. The first occurrence of the text after the cursor position is now highlighted. Another click on the Find button will move to the next occurrence.

If you want to replace a piece of text, just enter or copy the original text into the upper windowpane and the replacement text into the lower windowpane, place the cursor at the start position and click on the Find button. The first occurrence is now highlighted and can be replaced by selecting the Replace option from the menu under the Find button. Alternatively, if after the replacement you wish to search for another occurrence of the text, you could select the Replace&Find menu option.

If several replacements are required you can select the Replace All menu option to replace all occurrences of the text without further ado. However, this may not produce the result that is intended. For example, suppose you want to replace "some" with "many" and enter "some" in the upper pane and "many" in the lower pane. The find function just searches for "some", which means that "Some" is not found while "someone" is. Thus replacing all occurrences is a facility that should be used cautiously.

The keyboard functions, incremental-search and query-replace, can also be used for the same purpose. However, these functions can't be interrupted as easily as the find and replace functions. This is awkward because, while you are searching or replacing text, you often want to edit something else, necessitating the use of the keyboard and demanding the cessation of the keyboard-initiated search operation. In contrast, because the find-and-replace functions use the cursor position and the expressions in the two parts of the window and not the keyboard, interruptions are permitted without requiring any special procedure whatsoever. So, you can always edit the document (or consult another document, or do anything you like) between two find or replace actions. After you have finished editing, just reset the cursor to the position from which you want to continue searching and carry on with the find-and-replace action again.

Searching for and replacing operators is also possible. However, to find them all you have to search for them without their arguments. You can get an operator without arguments by copying only the operator, by clicking on the operator in a stencil palette and deleting the arguments or by using the Insert Op. console command and then clicking on the operator.

You can use the replace function to change all the identifiers of a document. To replace every " x " in a document with a " y ", enter " x " in the find window and " y " in the replace window. Put the cursor at the beginning of the document and click on the Replace All menu option. Now, every " x " is replaced by " y ". The difference between this replace and the rename command is the fact that bound " x " 's are also renamed. The normal query-replace is different too because that command will also change the "x" 's in words like "box" or "expression". Thus, if you type "x" (the TEX command for " x "), the replace function will not rename such an " x ", whereas the query-replace will. (The situation is not likely to occur. You will soon rid yourself of the habit of writing raw TEX or LATEX!)

If you want to frequently change large expressions according to a particular rule, it is convenient to define the rule in the find-and-replace window, save it, and use it repeatedly. For instance, the rule

$$a{\cdot}x^2 \;+\; b{\cdot}x \;+\; c \;=\; a \cdot (x + \frac{b+\sqrt{b^2 - 4{\cdot}a{\cdot}c}}{2{\cdot}a}) \cdot (x + \frac{b-\sqrt{b^2 - 4{\cdot}a{\cdot}c}}{2{\cdot}a})$$

can be entered as a find-and-replace operation. The upper pane should be set to contain

$$\mathbf{E}_1{\cdot}\mathbf{E}_4{}^2 \;+\; \mathbf{E}_2{\cdot}\mathbf{E}_4 \;+\; \mathbf{E}_3$$

while the lower pane should contain

$$\mathbf{E}_1 \cdot (\mathbf{E}_4 + \frac{\mathbf{E}_2 + \sqrt{\mathbf{E}_2{}^2 - 4 \cdot \mathbf{E}_1 \cdot \mathbf{E}_3}}{2 \cdot \mathbf{E}_1}) \cdot (\mathbf{E}_4 + \frac{\mathbf{E}_2 - \sqrt{\mathbf{E}_2{}^2 - 4 \cdot \mathbf{E}_1 \cdot \mathbf{E}_3}}{2 \cdot \mathbf{E}_1})$$

Now, you can rewrite simple expressions:

$$1 \cdot x^2 \ + \ -5 \cdot x \ + \ 6$$

becomes

$$1 \cdot (x + \frac{-5 + \sqrt{(-5)^2 - 4 \cdot 1 \cdot 6}}{2 \cdot 1}) \cdot (x + \frac{-5 - \sqrt{(-5)^2 - 4 \cdot 1 \cdot 6}}{2 \cdot 1})$$

and

$$(x+y) \cdot x^2 \ + \ y \cdot x \ + \ y$$

becomes

$$(x+y) \cdot (x + \frac{y + \sqrt{y^2 - 4 \cdot (x+y) \cdot y}}{2 \cdot (x+y)}) \cdot (x + \frac{y - \sqrt{y^2 - 4 \cdot (x+y) \cdot y}}{2 \cdot (x+y)})$$

As illustrated by these examples, Math∫pad has to match every part of the expression. The expression $1 \cdot x^2$ is completely different from x^2, although it is mathematically equal. Also $-5 \cdot x$ differs from $-\ 5 \cdot x$ (in the former 5 is negated, in the latter $5 \cdot x$). Such "knowledge" of arithmetic is not built into Math∫pad making it sometimes seemingly cumbersome to use in familiar areas of mathematics. Note, however, that Math∫pad can match subexpressions like $x+y$ as arguments and that parentheses and spacing are ignored in the matching process. When a replacement is made, the parentheses and spacing are added as specified in the replace expression. The find & replace can therefore be used to remove parentheses or add spacing.

Chapter 11

Stencils and Templates

Unlike most mathematics editors, Math∫pad has no built-in notational conventions — for very good reasons.

Firstly, notational conventions depend heavily on the research area. Many groups have their own conventions and would prefer to see their own favourites built in. An editor written to cater for such preferences can only be used easily by a limited group while other users are restricted by the defined conventions.

Secondly, the conventions used by an individual or group can alter during the course of time. A reason for a change can be the start of a new task or a technical development. Until a few years ago, for example, the best on-screen representation of a summation over an interval might have looked like

$$(\underline{\mathtt{S}}\ \mathtt{i}\ :\ \ 0 \leq \mathtt{i} \leq \mathtt{N}\ :\ \mathtt{i}^2)$$

whereas now, of course, it is possible to write directly

$$\sum_{i=0}^{N} i^2 \quad .$$

So changes may result from developments in resources and technology.

Thirdly, some operators have a different meaning and precedence depending on the context in which they are used. The "+" operator, for example, is used for many purposes: to denote addition in arithmetic (of course), the Boolean $\vee$ (or) operator, and the disjoint sum of two types, to name just a few. It may be that the same document employs the operator in different ways. Typically the precedence of the operator depends on its meaning and were such an operator to be built in, only one precedence could be used for it.

Fourthly, the diversity of conventions and operators is so large that it is impossible to build them all into the editor. Any good editor will provide a facility to expand on the existing conventions.

Instead of built-in conventions Math∫pad has a powerful facility to create new "stencils" or to change existing ones, so that every user can make his own set of notational conventions. Math∫pad recognises the fact that Mathematics is not a dead language but very much alive and kicking. Put simply, Math∫pad puts the "s" back into Mathematics!

A Math∫pad stencil consists of a set of "templates" each of which may have several "versions". In this chapter we discuss stencils and templates in general and assigning values to template-specific parameters. In chapter 12 we describe in detail how to define a version of a template.

11.1 Document Stencils and Code Numbers

For the most part, the user of Math∫pad need not be aware of how templates are stored or their definitions retrieved. However, some information is important, in order to avert mishaps.

To achieve portability of Math∫pad documents, the definition of all templates used in a document is an integral part of each document. Specifically, the complete definition of a template (including the definition of all versions of the template) is stored in a document's `mpd` file at the position of its first occurrence in the document. Subsequently, a code number indicates the use of a version of the template, this code number being unique to individual versions and assigned to each version at the time of its creation. The advantage of this process is that it is not necessary to transmit stencils along with document files when sending Math∫pad documents to a colleague. (It is necessary, however, to transmit any local LaTeX macros that have been used in the definition of a template.)

There is, of course, never any need for a user to know what any of the code numbers are (nor how they are generated) but it is important to realise that it is the code number that identifies a version of a template and not its definition. It is possible to create two templates whose definitions are identical in every detail but whose code numbers are different. The effect is that the system treats the two templates as different. Keep this in mind when, for example, searching for a template using the find & replace window.

The code number of the template rather than, say, its visual appearance on the screen is used in the search.

It is possible to replicate templates in different stencils. To see whether two templates from different stencils are identical (i.e. have the same code number) ensure both stencils are loaded and then inspect the "Info" associated with one of the templates. (See section 8.1.2.) If both stencil names appear under the "Files" entry in the Information menu then they are indeed identical, otherwise they are not. Take the document you are reading now as an example. Load the document stencil. In addition, load the **latex2e** stencil. Then inspect the information on, say, the "LaTeX" template. You will see that both `latex2e` and `StencilsandTemplates.mpd` are listed as files where the template can be found.

11.2 The Define Window

A template is defined in the *define* window an example of which is shown in figure 11.1. Reading from top to bottom, a define window has six elements: a header, five buttons, a command menu (including the kind, precedence and spacing definitions), a "Name", a "Help file", and the definition proper. The definition proper is further divided into a number of sections (separated by a horizontal line) each of which defines a version of the template. We discuss each element in turn.

11.2.1 The Header

A define window is opened by clicking on the Define button in a stencil window. The name of the stencil is entered in the header so that at all times the header indicates which stencil will be changed. Once the define window is open, another stencil can be chosen by clicking on its Define button. (If two stencils with the same name are loaded then the stencils are automatically numbered, the number being added in angle brackets after the name, so that it is still evident which stencil will be modified.)

11.2.2 Buttons

The buttons at the top of the define window have the following functions:

New Install a new template.

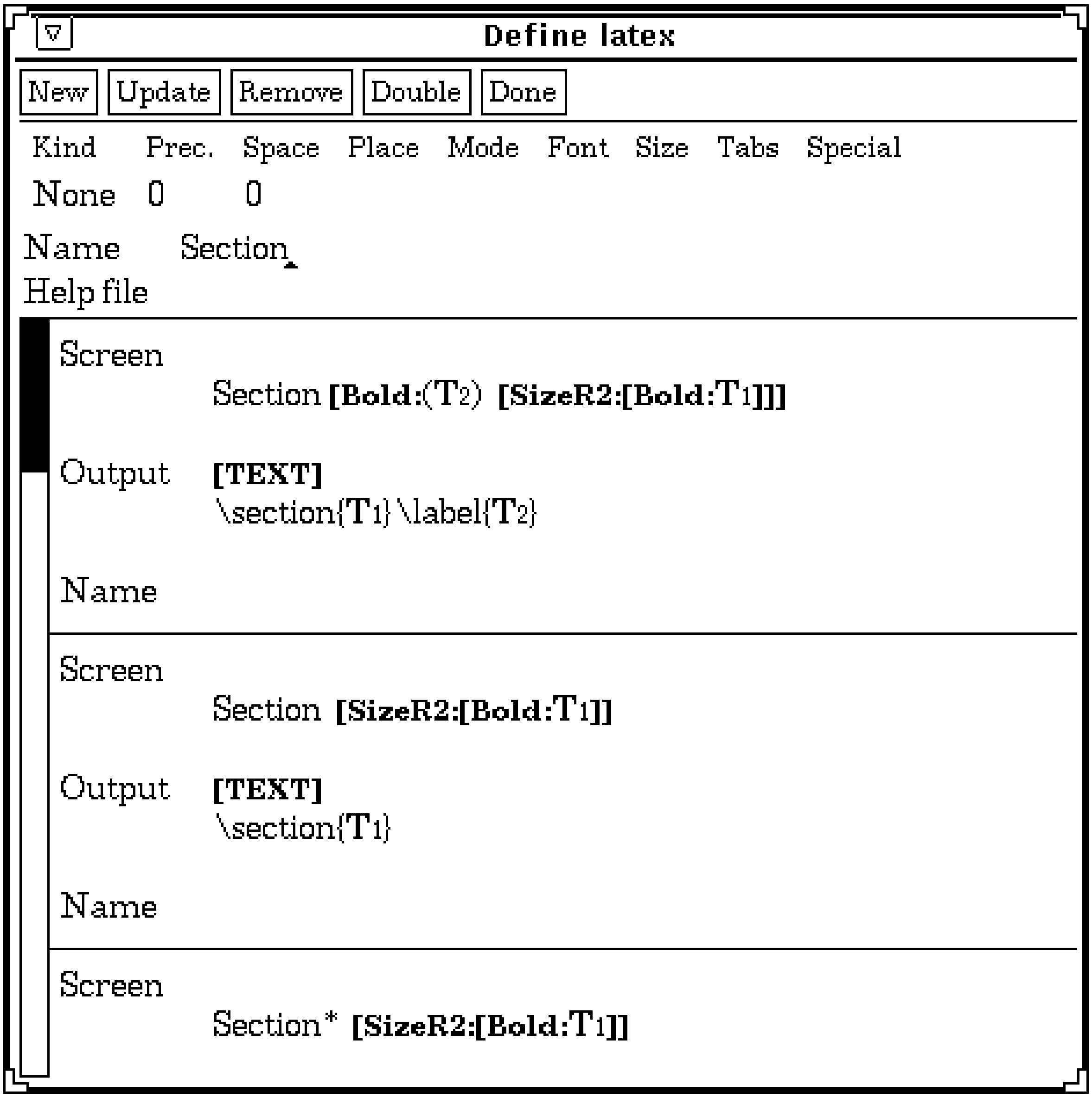


Figure 11.1: The Define Window

Update Update an existing template.

Remove Remove a template. The contents of the define window will not be removed so the template can be reinstalled if it was deleted by mistake. Use Update for reinstallation, not New so that the template retains the same code number. (See below).

Double This button is used to assign identical code numbers to templates that are identical but for their code numbers. It is also used to remove multiple copies of identical templates.

The template in the define window is compared with all other templates (in loaded stencils). If it is identical to another in every respect but for its code number, identify the two by modifying the code number assigned to the template in the define window. The template in the define window will then occur "doubly" (and possibly multiply); that is, it occurs in the current stencil and in one or more other stencils. (Note that a template cannot occur doubly in the same stencil.) If the control key is held down at the same time as Double then the doubly occurring template in the current stencil will be removed.

If the define window is empty, the operation will compare *all* templates in the stencil with templates in other stencils. By the same token, if the control key is held down then all doubly occurring templates in the current stencil will be removed.

Done Close the define window. The template in the define window will be removed, so the template should be installed first if necessary.

The New button has the effect of creating an entirely new template (even if the contents of the define window is an exact copy of an existing template). A new code number is created for each version in the template. Update, in contrast, preserves existing code numbers, new code numbers being generated only for newly added versions.

Update will destroy the existing contents of a stencil. Thus when experimenting with the definition of a template it may be advisable to use New at first. See section 12.9 for the procedure by which to do this.

To copy a template from one stencil to another proceed as follows: Click on Define in the stencil to which the template is to be added. Click on the template to be copied. (The template's definition will appear in the define window.) Finally click on Update in the define window.

11.2.3 Command Menu

The on-screen layout and composition of templates is specified by means of a simple command language. Creating terms in this language is menu-driven. Below the buttons of the define window there is a list of items used to select the various commands. (See the table below.) By clicking on an item, a pin-up menu will appear and a choice of command can be made. In this section we limit ourselves to a brief description of each item and give a reference to where more detailed information can be found.

The items are as follows:

Kind The kind of a template is selected here. Templates can be (unary or binary) mathematical operators or not. In the case that a template specifies a mathematical operator the kind is further subdivided into prefix, postfix, infix, left associative or right associative operator. Other templates, whether a mathematical operator of higher arity or non-mathematical, have kind None. (See section 11.3.2.)

Prec. The precedence of an operator can be selected from the pin-up menu. The selected precedence is displayed immediately below this item. The precedence of a template with kind None is irrelevant and will be set to zero when the template is installed. (See section 11.3.3.)

Space The minimum space to be inserted around an operator is set with this button. The selected width is the product of the chosen number and the unit width specified in the default window. This information is ignored if the kind is None. (See section 11.3.4.)

Place The different types of place holder.

Mode Commands specifying the LaTeX mode. (See section 12.4.)

Font The fonts available (at the time execution of Math∫pad was started). (See section 12.5.)

Size The different font sizes Math∫pad can handle. (See section 12.6.)

Tabs Tabbing commands. (See section 12.7.)

Special A number of special constructions, such as vertical stacking and adding elastic space around templates. (See section 12.8.)

11.2.4 Template Specification Window

The major portion of the define window is taken up by the windowpane (recognisable by the scroll bar at the left side). We shall call the contents of the windowpane the *template specification*. When the define window is first opened the windowpane is entirely clear. Certain edit operations cause the windowpane to be divided into sections, each section containing three fields labelled "Screen", "Output" and "Name". Each section defines a version of the template. We call each section a *version specification*.

The "Name" field in a version specification is used in the pin-up menu that lists the versions of a template that are available. If a name is not entered, Math∫pad will invent one based on the stripped screen format.

The "Screen" field normally displays a term in a screen format command language indicating how the version is to be displayed on the screen. The "Output" field determines the output that is to be produced by Math∫pad when the Output button at the head of a Math∫pad document window is selected.

11.3 Defining Templates

To add a new template to a stencil or to amend an existing template click on the Define button at the top of the appropriate stencil. The define window will appear. If a new template is to be defined one must first enter the template specific data. This is discussed in the following subsections. Subsequently a definition for each version of the template must be specified. How to define a new or amend an existing version is described in chapter 12.

An existing template can be used as a basis for a new one as follows: Click on the Define button at the top of the stencil to which the new template is to be added. Click on the template to be used as a model and its definition will appear in the define window. Edit the template specific data and version specifications as required and then click on the New button in the define window.

11.3.1 Naming a Template

A template can be named by typing in the desired name in the Name field. This name may only contain keyboard characters and it will be used to identify the template in the stencil. When a define window is opened the cursor is automatically set in the

Name field. The cursor can be reset by clicking with the near mouse button in the desired place. If the name of the template is left empty, the name or stripped screen format of its first version will be used to represent the template in the stencil. See chapter 12 for information on naming versions.

11.3.2 A Template's Kind

The next step in defining a new template is typically to specify its *kind*. The kind of a template stipulates whether the template is or isn't a unary or binary mathematical operator. Unary and binary operators are further subdivided into prefix, postfix, infix, left associative and right associative operators. Non-mathematical templates and mathematical operators of higher arity have kind None.

So far as Math∫pad is concerned a unary operator (prefix or postfix) is a template having a single implicit expression place holder that is additional to any place holders that appear explicitly in the definition of (versions of) the template. This implicit place holder is called the argument of the operator and is furthermore the principal expression place holder of the template. A *prefix* operator is an operator that is written before its argument. Examples are the minus operator as in -2 and the logarithm operator as in $\ln x$. A *postfix* operator is an operator that is written after its argument.

Again so far as Math∫pad is concerned, a binary operator is a template having two implicit expression place holders in addition to any place holders that appear explicitly in the definition of (versions of) the template. These implicit place holders are called the arguments of the operator and are positioned immediately to the left and right of the operator. The leftmost argument is the principal expression place holder of the template. Binary operators can be either "infix", "left associative" or "right associative". "Infix" operators are used to form continued expressions like $0 \leq i < N$ and $a{+}b{-}c$, "left associative" operators and "right associative" operators are exemplified by the Boolean "follows from" and "implies" operators (as in $a{\Leftarrow}b{\Leftarrow}c$ meaning $(a{\Leftarrow}b){\Leftarrow}c$, and $p{\Rightarrow}q{\Rightarrow}r$ meaning $p{\Rightarrow}(q{\Rightarrow}r)$), respectively.

The structure of expressions constructed using "infix" operators is much richer (in the sense that they have many more subexpressions) than that of expressions constructed using left associative and right associative operators (see chapter 9 for more details) and we recommend that you do not use the latter two categories.

The designation None refers to the fact that a template has no implicit expression place holders.

Note that operators may be quite complicated; they may include additional place holders, or may involve complicated layout rules. You may wish, for example, to specify an infix arrow operator, "$\rightarrow$", with one version having an additional expression place holder written as a superscript to the arrow, as in $a \rightarrow^{f} b$, and a second version in which the two arguments are written on separate lines, one below the other. How this can be achieved is explained in chapter 12.

It is vital to note that kind is associated with the template itself and not with a version of the template. Thus it is not possible to define, for example, a template for a unary operator one version of which is a prefix operator and the other a postfix operator. The precedence and spacing fields are also template, not version, specific and are only relevant for templates of kind other than None.

Unary and Binary Operators

To define a unary or binary mathematical operator, choose one of the options Prefix, Postfix, Infix, Left and Right from the Kind menu. (Left and Right are short for "Left associative" and "Right associative" infix operator, respectively.) Specify in addition its precedence (section 11.3.3), and, if desired, the minimum spacing between the operator and its argument(s) (section 11.3.4). Finally follow the instructions in chapter 12 in order to specify how (each version of) the operator is to be denoted.

The code generated by Math∫pad is such that TEX conventions for spacing mathematical expressions are ignored. The spacing generated by Math∫pad depends on the position of the operator in the expression tree (the principal operator in an expression being surrounded by the most space in the absence of parentheses).

The place holders for the arguments of an operator (one in the case of a prefix or postfix operator and two in the case of an infix operator) are generated by the system when an operator is entered into an expression place holder in a document and are not shown in the version specification. Only the operator itself is shown.

Non-mathematical Templates and Multi-fix Operators

To define a non-mathematical template or multi-fix operator choose the option None from the Kind menu. Then follow the instructions in chapter 12 in order to specify how (each version of) the operator is to be denoted.

11.3.3 Precedence of a Template

The precedence of a template is specified by a number between zero and twenty where the higher the number the higher the precedence. A new template is automatically given a zero value and this can be altered by choosing a number from the menu under Prec. in the command menu. The value is displayed underneath Prec. The precedence is ignored for templates of kind None.

We recommend that unary operators are given the highest precedence. For binary operators a good guide is the smaller the size of the symbol the higher the precedence. If this is not the case we suggest that the template is given a non-zero minimum spacing. (See section 11.3.4.)

11.3.4 Spacing around a Template

The minimum spacing around a template is by default zero. This can be altered by selecting a number between zero and twenty from the menu under Space. The minimum spacing is increased by this number times the unit length of elastic spacing (defined in the default window). The current value of the number is displayed under Space and is only relevant for templates of kind other than None. Thus if the number 2 is chosen and the unit length of elastic space is 3 pixels then a space at least 6 pixels wide will be inserted at each side of a binary operator.

We recommend that all unary and binary operators should have a minimum spacing of zero except for binary operators with a low precedence which are represented by a small symbol.

Remember that a template of a unary or binary mathematical operator also has elastic space between the operator and its argument(s). This elastic space grows with the complexity of its arguments, that is the depth of the expression tree. Templates of kind None do not have this. To add elastic space around a multifix operator with kind None see section 12.8.2.

11.3.5 The Help File

It is possible to associate a Math∫pad document with a template in order to provide assistance or guidance in the use of the template. Information about where to find the help information should be entered in the "Help File" field of the define window. The information to be provided consists of two items separated by the symbol "#".

The first item is the name of a Math∫pad document (minus the "`.mpd`" extension), and the second item is a string that is used to identify the position in the document where the description of the template begins. The environment variable `MATHPADHELPPATH` is a colon-separated list of directories in which to search for help files. Its function is comparable with that of the standard Unix PATH variable.

For example, the html stencil is supplied with the help file called `html.mpd` and stored in the `lib` directory. Inspection of, say, the definition of the Glossary template in the html stencil reveals the entry "html#glossary" in the "Help File" field. Assuming that `MATHPADHELPPATH` is correctly set, requesting help on the Glossary template will cause the `html.mpd` document to be loaded and the cursor to be set in that document to the first occurrence of a text place holder filled by the string "glossary".

The file name may include a path indicating where to find the help file relative to the value of `MATHPADHELPPATH`. An example is

manual/latex#FontHelp

indicating that the help document `latex.mpd` is located in a directory `manual`, which is a subdirectory of one of the directories specified by `MATHPADHELPPATH`. The location of the help information specific to the template is at the first occurrence of a text place holder containing the word "FontHelp".

The entry in the help field may be prefixed with the symbol "$\sim$" or an environment variable. Examples are

~smith/HelpDirectory/ExStencil#ExTemplate

indicating that the help document `ExStencil.mpd` is located in user smith's `HelpDirectory`, and

$HOME/HelpDirectory/My1stStencil#My1stTemplate

indicating that the help document `My1stStencil.mpd` is located in the subdirectory `HelpDirectory` of the user's home directory.

The string following the "#" symbol must be the entire contents of a text place holder (rather than a substring of a portion of text). This restriction is designed to improve the efficiency of the search process and to reduce the possibility that a change in the help file has the unintended side effect of altering the position in the file associated with a template. It is recommended that you use a special template to insert such links in your document and you use a special convention to ensure that the names of links do not coincide with the contents of other text place holders (such as the labels given to equations).

11.3.6 Requirements

There are a few requirements on the definition of templates, specifically:

- There is at least one version.
- Either the template has a name, the first version has a name or the first version has a visible screen format. (Spaces, tabs and return characters are all counted as non-visible.) This check ensures that the template can be selected after it is installed.
- No screen format contains repeated occurrences of a place holder.
- There is at least one screen format that contains *all* the place holders in all versions of the template.

When any of these requirements is violated, the cursor will be placed at the position where the first violation occurs, and the installation of the template will abort.

If a place holder occurring in the output field does not occur in the corresponding screen field a warning message appears. The installation will nevertheless take place. Such place holders are called *hidden* place holders because the value assigned to the place holder and used in the generation of output is hidden from view. The final requirement on place holders guarantees that hidden place holders can be edited (by switching to the version in which all place holders are visible). The use of hidden place holders is a very convenient way of improving the on-screen readability of documents. (It is akin to the "folding facility" often built into structure editors.

11.3.7 Rearranging Versions and Templates

The commands shown in table 11.1 are used to reorganise versions of a template. For them to take effect the cursor should be placed anywhere in the "Screen", "Output" or "Name" fields of the version that is to be repositioned.

Templates can also be rearranged within a stencil. (See table 11.2.) To move a template open the define window and select the desired template so that its definition appears in the define window. Then, with the cursor in the "Name" field of the template, the following keys can be used to move the template to the left or right in the stencil, or to the first or last position in the relevant section of the template. (An attempt to move a template out of, say, the "None" section into another section will of course fail.)

Key	Action
M-up	Move the version up one position.
M-down	Move the version down one position.
M-home	Move the version to the first position.
M-end	Move the version to the last position.

Table 11.1: Moving versions within a template.

Key	Action
M-Left	Move the template one position to the left.
M-Right	Move the template one position to the right.
M-PgUp	Move the template to the first position.
M-PgDn	Move the template to the last position.

Table 11.2: Moving templates within a stencil.

Chapter 12

Version Definition

12.1 Introduction

In this chapter we describe in detail how to define a version of a template. As mentioned in section 11.2.4 this is done in a section of the define window's windowpane called a version specification. Each version of the template has its own version specification section.

The version specification has three fields. These are labelled "Screen", "Output" and "Name". The "Name" field is used in the pin-up menu that lists the versions of a template that are available and may be left blank. In this case Math∫pad will invent a name based on the screen format. This usually suffices for simple cases but quickly becomes inadequate.

The "Screen" field normally displays a term in a screen format command language indicating how the version is to be displayed on the screen. The "Output" field determines the output that is to be produced by Math∫pad when the Output button at the head of a document window is selected. Built into Math∫pad is a mechanism for automatically generating LaTeX code corresponding to the screen version but the user is free to override this mechanism so causing Math∫pad to generate code for any ascii-based markup language.

The screen format command language should be reasonably self-explanatory. Statements of this language are displayed in a small bold font whilst user input is displayed in the default font. Most of the commands consist of a single token made up of the menu item used to enter the command bracketed by the symbols "[" and "]". Examples are **[tab]** and **[Line]**. Table 12.1 lists the commands that use multiple tokens in order to define their scope or to separate their fields.

Command	Tokens
Font change	[*name*:] where *name* is the name of a loaded font.
Size change	**[SizeR*n*:]** where *n* is the relative size change.
Stack Base	**[StackB: : :]**
Stack Centre	**[StackC: : :]**
Tabbing	**[Tabbing:]**
Display	**[Display:]**

Table 12.1: Multiple-token screen format commands.

Note that whereas the closing braces all look the same they are in fact different. Math∫pad will ensure that the different braces are used in matching pairs and are nested correctly. The "**R**" in the size change command stands for "Relative". (An earlier implementation had an absolute size change command which may be reinstated sometime in the future.)

The small bold font used in the "Screen" field is also used in the "Output" field to distinguish system commands from user input.

12.2 Editing a Version

Before one can amend an existing version of, or add a new version to, a template one must first open the define window and then either load the template or, in the case of a new template, set its template-specific parameters. (See section 11.3.) To edit a field of an existing version click in that field and change it as required. If a new version is to be added to a template click in the template specification window below the last version specification or anywhere in the windowpane if there are no existing versions. A copy of the first version specification, or if there is no existing version, an empty specification will be inserted. As many versions as you like may be added in this way.

A version can be removed from a template by setting the cursor anywhere in the version and executing the keyboard command **M-Del**. There is no menu item for this operation.

A version specification is constructed by selecting entries from the command menu at the head of the define window and by entering text from the keyboard. Typically, information will be entered simultaneously in both the "Screen" and "Output" fields and two cursors will be visible. To modify only the automatically generated LaTeX code click anywhere in the "Output" field. The cursor in the "Screen" field will disappear. Any text entered from the keyboard or command entered from the menu will then modify the "Output" field but not alter the screen format.

It is possible to edit the screen version in two modes. The first mode displays command tokens and does not interpret them. The second mode interprets the tokens making them no longer visible. That is, it shows the format of the version as it will actually appear on the screen. The first mode is used to edit the screen version whilst the second mode is used to preview the version. The **M-T** key switches between these two modes.

Several of the basic text editing commands (see chapter 7) are available for use, although not all. (See the list of entries labelled "Define" in the `mpk` file that you are using for precise details.) Two that are available are **C-A** (move to the beginning of a line) and **C-K** (kill a line). Our advice is to just try what you would normally do in a document window. It may happen that nothing occurs but, in any case, no harm will be done.

The multiple-token commands get special treatment in order to ensure that braces remain properly matched. In particular, if you remove one of the tokens of such a command all the other tokens are also removed. For example, if you have the construction "**[Bold:**Section**]**" and you remove the "**]**", you obtain "Section". Similarly, removing a "**:**" from "**[StackC:[Line]:[Bar]**Box**[Bar]:[Line]]**" results in "**[Line][Bar]**Box**[Bar][Line]**".

The transpose function (**C-T**) is particularly convenient for editing multiple-token commands. It works by moving the tokens to the left and right of the cursor according to the following rules (in the order given):

- If one of the two tokens adjacent to the cursor is not part of a multi-token command, the transpose will swap the two tokens and move the cursor one position forward (just like the normal transpose).

- If the right token is an opening brace, the left token and cursor are moved to after the matching closing brace.

- If the left token is a closing brace, the right token will be moved before the matching opening brace.

- Otherwise, no tokens are moved and the cursor is moved one position forward.

Table 12.2 illustrates the use of transposition. A small diamond is used in the table to show the position of the cursor.

Before	After transposing	Rule
[**Bold:**]$_\diamond$word	[**Bold:**w]$_\diamond$ord	1
[**SizeR1:**]$_\diamond$[**Bold:**Section]	[**SizeR1:**[**Bold:**Section]]$_\diamond$	2
[**StackB:**-::$_\diamond$[**Italic:T**1]]	[**StackB:**-:[**Italic:T**1]:$_\diamond$]	2
[**SizeR-1:**$_\diamond$[**StackC:**::**E**1]]	[**StackC:**::**E**1][**SizeR-1:**$_\diamond$]	2
[**SizeR1:**[**Bold:**Section]$_\diamond$]	[**SizeR1:**][**Bold:**Section]$_\diamond$	3
[**Italic:**$_\diamond$]It	[**Italic:**]$_\diamond$It	4

Table 12.2: Examples of transposition.

12.3 Place Holders

When a version of a template is defined, ***place holders*** are used to show the positions of the arguments in the version. (Remember that the place holders for the arguments of a unary or binary operator, one or two respectively, are generated by the system when such an operator is entered into an expression place holder in a document and should not be added to the version specification.) To enter a place holder into a version specification select the required type from the pin-up menu which appears when Place at the top of the window is selected. The available options are listed below together with the symbol used to represent them on the screen.

E, expression
: An expression, replaceable by a template, an operator sequence or an identifier. The expression will be set in the math mode of LaTeX.

O, operator
: An operator, replaceable by a template or an expression. An operator will be set in math mode unless the template specifies that it should be placed in text mode. The spacing that TeX adds around operators is overridden by Math∫pad's own spacing conventions.

I, identifier
: An identifier, which should not be too long and can only contain characters from the keyboard or the symbol palette. An identifier can't contain a space or a newline. By default, identifiers are set in LaTeX's math italic font. (Users of LaTeX version 2.09 should insert a suitable definition of `\mathit` in their macro file.)

V, variable
: A list of bound identifiers separated by commas. Variable place holders are used to specify the scope of identifier renaming. (See section 9.5.) Otherwise variables and identifiers are treated the same. In particular, variables are also set by default in math italic font.

T, text A text containing any combination of text and expressions. Text will be set in text mode and LaTeX will handle the line and page breaks. However, line breaking inside the tabbing environment must be done by the user.

Name This option is not itself a place holder but is used to give a name to a place holder. See below for further details.

Place holders can be inserted by selecting them from the pin-up menu or by using the keyboard shortcuts (**CM-E**, **CM-O**, **CM-I**, **CM-V**and **CM-T**).

The final entry in the Place menu, "**Name**", is used to label place holders. In order to name a place holder, the token [**Name:**] should be entered into the version definition immediately to the left of the place holder to which it applies. Then fill in the required name. When an instance of the template is entered into a document, the default name of the empty place holder ("**E**", "**O**" etc., as the case may be) will be replaced by the chosen name. The name given to a place holder in the screen section and the output section may differ. This possibility is rarely used but can be useful in giving a default value to a place holder.

The use of named placed holders does *not* place any additional restriction on the use of the place holder. Naming a place holder "Statement", for example, will not mean that only versions of a Statement template may be entered into the place holder.

The LaTeX output that is produced by an empty place holder in a document is the command "`\mpT`" in the case of a text place holder, "`\mpE`" in the case of an expression place holder, and so on. In the case of a named place holder the command "`\mpN`" is followed by an argument which is the name given to the place holder. For example, an unfilled place holder named "**color**" would cause "`\mpN{color}`" to be output. These commands are defined in the file `mathpad.sty` distributed with the system (so that they generate the same bold characters) and can be redefined to meet your own requirements.

Each place holder entered will be given a new index. (But see below for how to stipulate which index should be used.) The maximum number of place holders allowed in any one template is fifteen. The indices assigned to place holders are used for the following purposes:

(a) to identify the "principal place holder" of a given type in a template,

(b) to relate place holders in different versions of a template,

(c) to relate the screen format and the LaTeX output of individual versions of a template.

The principal place holder of a given type is used to determine which place holder is occupied by the target term when a template is selected. For example, suppose a "quantifier" template of the form $\forall(\mathbf{V}{:}\mathbf{E}{:}\mathbf{E})$ is defined. Suppose also that the target is set to the expression $x = y$. Then, if the principal expression place holder is defined to be the rightmost place holder, clicking on the quantifier template will result in the expression $\forall(\mathbf{V}\colon \mathbf{E}\colon x = y)$. If, on the other hand, the leftmost expression place holder is the principal place holder then the expression $\forall(\mathbf{V}\colon x = y\colon \mathbf{E})$ will be the outcome.

The principal expression place holder of unary or binary operators (templates of kind other than None) is predetermined. For unary operators it is its argument and for binary operators it is the left argument. Otherwise the principal place holder of a given type is the place holder of that type with the lowest index. The automatic mechanism for generating indices to place holders will always result in the leftmost place holder (of a given type) becoming the principal holder. We describe shortly how to override this mechanism.

The indices also determine the movement of the cursor once a place holder has been filled in. Specifically the cursor moves to the lowest numbered place holder not already filled.

The screen format may not contain any place holder twice. This requirement is normally automatically fulfilled because the system generates a new index each time a place holder is inserted. In the LaTeX code, however, a place holder can be used several times. This is useful at positions where output should be copied.

When switching between versions of a template the index information is used to determine how the place holders are to be filled in. In this way the order of occurrence of arguments can vary from one version to another. As an example one may wish to define a template with two versions, a "let" statement and a "where" statement, both having two arguments. Further it should be the case that switching version from, say,

> **let** $a = b+c$ **in**
> **begin**
> $\quad x := a$
> **end**

gives rise to

> **begin**
> $\quad x := a$
> **end**
> **where** $a = b+c$.

This is easily achieved by stipulating that the first placeholder in a "let" statement has index 1 whereas the place holder with index 1 in a "where" statement is the second one. Similarly, the second place holder in a "let" statement is stipulated to have index 2 whereas the place holder with index 2 in a "where" statement is the first one.

To stipulate the index to be assigned to a place holder the **C-U** command should be used with the correct index. For example,

C-U 3 **CM-E**

will insert an expression place holder with index 3. Such a command will always be successful if only applied to the LaTeX output. However the stipulated index will be ignored and the automatic index generation invoked if the command would result in a duplicate of the place holder in the screen format. Note also that there is one index count used for all the different types of place holder. Thus introduction of an expression place holder with index 3, say, will be refused if there is already a text place holder with index 3. (This may alter in future versions of the system.)

Example 12.1 Many LaTeX documents start with very similar preambles. Often only the title and packages required are different. A simple template with place holders for the title and names of packages can be used to insert the complete preamble. For example:

Screen: Article Packages:**T**2 Title: **T**1
Output: [**TEXT**]\documentclass{article}
\usepackage{mathpad,**T**2}
\input{mpmacros.tex}
\title{**T**1}\author{<Your name>}
\maketitle\begin{document}
Name: Preamble

Note that the template has two text place holders the second of which has the lower index. The second place holder is thus the principal place holder. Entering a text, for example "My Latest Report", making that text the target term and then selecting the template from the stencil window will enter the text into the title place holder in the template.

With the aid of features yet to be described, the title can be set on the screen in a large bold font. A version can also be made in which the packages (entered in the field labelled "Packages") are not visible on the screen but are entered in the LaTeX output. Additional arguments can be added and versions can be made to switch between articles, reports, books and other styles. With such a template, the commands you would otherwise type or copy into every document (your name, your address etc.) are now inserted by a simple click. The variable information is directly visible but other low-level "noise" (e.g. the LaTeX commands) is not.
□

12.4 Modes

The topic of this section is very specific to generating LaTeX output. If you are using Math∫pad to generate terms in some other mark-up language (for example html) then there is just one thing you need to know and then you may ignore the rest of this section. Users of Math∫pad not generating LaTeX output must ensure that the "Output" field of each version specification begins with the symbol "[**TEXT**]". This instructs the system to treat what follows in the "Output" field as plain text and not to insert "mode" commands specific to TeX and LaTeX. (In normal circumstances ensuring that this is so entails doing nothing. If, however, you define the kind of a template to be anything other than None then the "Output" field of each version will by default begin with the symbol "[**MATH**]". In this case select "Toggle" under the Mode button repeatedly until the symbol "[**TEXT**]" appears.)

TEX uses different modes to process the input. The main modes are LR mode, paragraph mode and math mode. The first two modes are used for normal text while the math mode is used for mathematical expressions and symbols. In paragraph mode TEX will automatically determine line and page breaks, in LR mode this action is turned off.

The commands for changing mode in TEX do not have a systematic structure causing some technical problems when using Math∫pad. Normally, the `$` sign can be used to switch between text mode and math mode but some other commands can also be used. Changing to math mode is compulsory for many symbols —the so-called "Mathematical symbols" which includes Greek letters like " α " but not (for example) the symbol "£"— even when the symbol occurs on its own in plain text. The `\mbox` command is used to change to LR mode, regardless of the mode it is used in. The `$$` command however will cause a change to displayed math mode if used in text mode, will have no effect if used in math mode and will cause a change to text mode if used in displayed math mode.

Math∫pad will insert mode changing commands as appropriate. To do this, `$` signs are used to switch between text mode and math mode while the `\mbox{}` command is used to switch between displayed math mode and text mode.

The "Output" field of each version specification must begin with one of the three output mode commands TEXT, MATH or BOTH. The command cannot be removed or inserted at any other position. MATH directs Math∫pad to insert mode changing commands where necessary to ensure that the contents of the template version are set in math mode. TEXT behaves similarly. BOTH means that the template version may be used in either text or math mode so that mode switching commands are unnecessary.

Note that mode changing commands inserted by the user in a Math∫pad document can cause incorrect LATEX to be generated. Care must be taken when mixing raw LATEX with Math∫pad templates in a document.

Built into Math∫pad is the assumption that templates of kind None will be set in text mode and all other templates in math mode. Therefore TEXT is automatically generated for templates of kind None and MATH for all other templates. The Toggle option under the Mode item allows you to override this assumption.

It is sometimes desirable to define templates that themselves change the mode. This can upset the automatic mode changing mechanism built into Math∫pad resulting in incorrect LATEX code. To remedy this, mode commands can be inserted into a template directing Math∫pad to use a particular mode rather than the mode that would automatically be chosen. The pin-up menu shown in table 12.3 appears when the Mode item in the define window is selected and is used to insert mode commands. The commands have the following definitions:

Mode
Toggle
text
math
display

Table 12.3: The Mode pin-up menu.

Toggle This button is used to switch ("toggle") between the three output modes: TEXT, MATH and BOTH. Each "Output" field must begin with one of these three modes.

text Set what follows assuming that the current mode is text mode.

math Set what follows assuming that the current mode is math mode.

display Set what follows assuming that the current mode is displayed math mode. Text entered in this mode will be put in an `\mbox{}` command. The remainder will be set in math mode.

A mode command should not be used to remove the $ signs added automatically by Math∫pad because such signs do not always appear in the LaTeX output. The $ signs around a place holder will only be inserted where necessary.

The need for mode commands is best explained with the aid of a number of examples.

12.4.1 Using TEXT and MATH

In LaTeX the macro `\underline` causes its argument to be underlined. The macro may be used in any mode and may therefore be used to underline both text and mathematical expressions. In order to underline text or expressions in Math∫pad the "stack" construction is used (described in section 12.8). The output that is generated must however take account of whether the argument is text or an expression. It is thus necessary to define a template with two versions as shown below. In the first version the mode command [**TEXT**] indicates that no $ symbols should be generated by the system.

Screen: [**StackB::T1:**[**Line**]]
Output: [**TEXT**]\underline{**T**1}
Name: underline (text)

(No special action is needed to insert "[**TEXT**]" in the "Output" field since this is the default when the kind of a template is None.)

In the second version \$ symbols should be generated (where necessary). The mode is thus toggled to MATH (indicated by the replacement of "[**TEXT**]" by "[**MATH**]" in the "Output" field.

Screen: [**StackB::E2:**[**Line**]]
Output: [**MATH**]**\$**\underline{**E**2}**\$**
Name: underline (math)

(Note that the (small, bold) **\$** symbols in the "Output" field are added automatically indicating to the user where Math∫pad will insert them if the version is used in inline text. That they are small and bold indicates that they have been inserted automatically rather than by hand.)

In principle only the first version is needed in Math∫pad since a text place holder can always be filled by an expression. If this version is used to construct, say, the expression

$$\underline{\ A\ } + \underline{\ B\ }$$

then the LaTeX that is generated by the system is as follows:

```
\underline{$A$}${+}$\underline{$B$}
```

If the second version is used then the LaTeX that is generated by the system is:

```
$\underline{A}{+}\underline{B}$
```

On screen $\underline{\ A\ } + \underline{\ B\ }$ and $\underline{A}{+}\underline{B}$ look identical. In the printed text, however, they are different. In addition to underlining A and B, a small amount of space around these two symbols is underlined in the first expression. The space around the addition operator is also larger in the first expression. The underlined space is called the "mathsurround" and is introduced automatically by TeX, the width being specified by the variable

`\mathsurround`. The signal to TEX that space should be inserted is an opening or closing $. The default value of the mathsurround is zero, so if you don't change it you won't experience this problem (but you may experience other problems of which we are unaware). If you do want a non-zero mathsurround then the quick and simple solution is thus to use the first version of the template for text arguments and the second for expression arguments.

12.4.2 Using BOTH and text

The BOTH command is used in situations where LATEX macros that can be used in either math or paragraph mode replace the system generated output. Typically, such commands switch the mode to LR mode. The box macros `\mbox` and `\fbox` are examples.

Suppose boxed items, such as $\boxed{a} - \boxed{b+c}$, are required in mathematical expressions. To enter such expressions, a template with one argument of type expression which frames the expression in a box is needed. Math∫pad does not have a boxing primitive but the effect can be simulated on screen using the Line and Bar primitives described in section 12.8. LATEX, on the other hand, does have a boxing primitive, namely `\fbox`. Rather than use the LATEX code constructed by Math∫pad it is preferable to edit the "Output" field by hand so that a call of `\fbox` is generated. Since an fbox can be used in either math or paragraph mode the output mode should be toggled to BOTH. In order to keep the output generation on the right track it is then necessary to inform Math∫pad that the argument of an `\fbox` command is processed by LATEX in LR mode. This is done by adding the hint text immediately before the expression place holder in the "Output" field as shown below.

Screen: **[StackB:[Line]:[Bar]E**1**[Bar]:[Line]]**
Output: **[BOTH]**\fbox{**[text]$E**1**$**}
Name: math fbox

In more detail the template is created as follows. First the mode is toggled to BOTH. Then the screen layout is created using the line and bar features described in section 12.8. Code will be automatically generated in the "Output" field but, on completing the description of the screen layout, this is erased by setting the cursor to the beginning of the "Output" field and killing the line (with **C-K**). "\fbox{" is then entered and the text entry in the Mode menu selected. Next an expression place holder is chosen from the Place menu, and finally "}"is entered.

12.4.3 Using math

The math mode command has a similar function to the text mode command. It is used in situations where manually inserted LaTeX macros that switch the mode to expression mode are to be used in text.

A simple example would be if you want in-line mathematical expressions in your text to be parenthesised by "\(" and "\)" rather than $ signs. The following template definition can be used for that purpose. The manually inserted "Output" definition indicates that the argument is to be set in text mode but that following the "\(" LaTeX will have switched to math mode.

Screen: **E**1
Output: [**TEXT**]\(\[**math**\]**E**1\)
Name: inline math

12.4.4 Using display

The final option in the Mode menu is labelled display. Its function is almost identical to that of the math option. It also indicates to Math∫pad that a LaTeX macro which switches the mode to math mode has been manually inserted in the "Output" field. The display option is used, however, in a math mode environment in which the text to be included is to be set within a LaTeX mbox. Here is an example.

Screen:
[**tab**] **E**1 **T**2

Output: [**TEXT**]\begin{equation}
[**disp**]**E**1~~~**\mbox{T**2**}**
\end{equation}
Name: Exp Txt

The purpose of the template is to create equations, each equation being concluded by text (perhaps a punctuation symbol like a period or a comma). The template must be set in text mode (hence the [**TEXT**] command) but following the "\begin{equation}" LaTeX will have switched to math mode. This is declared by the [**disp**] command. The effect of using [**disp**] rather than [**math**] is that any text place holders are automatically parenthesised by the mbox macro as indicated by the (system inserted) "\ **mbox{**" and "**}**" strings. (Had [**math**] been used instead then a **$** sign is inserted by the system to indicate that this is how the math mode will be terminated.) You are advised to use the display option rather than the math option if your template contains text arguments.

12.5 Fonts

Most texts use a wide variety of fonts — bold fonts for headings, italic fonts for emphasis and so on. Mathematical texts tend to use more than their fair share because judicious use of fonts can considerably enhance readability. (Unbridled use of different fonts can, however, be distracting.) Math∫pad allows unlimited use of different fonts both in text and in mathematical expressions.

Which fonts are available depends on the local installation but users can add their own fonts. (For further information on how to do so see chapter 14.) The file `fonts.mpt` is loaded when Math∫pad is started and describes which fonts are available and how they should be converted to LaTeX. These fonts are listed in the pin-up menu that appears when the Font item in the define window is selected. The `fonts.mpt` file provided in the standard distribution causes the pin-up menu shown in table 12.4 to be created.

Font
Roman
Bold
Symbol
Italic
MathSymbol
Bold Italic
Slanted
Sans Serif
Typewriter
CMSY

Table 12.4: The Font pin-up menu.

A font is selected by clicking on it. Two special symbols are then entered at the position of the cursor: one to open the specific font and one to close it. For the italic font, these symbols are [**Italic:** and] in the screen format and `{\it` and `\/}` in the LaTeX code (as specified in the fonts.mpt file). These two symbols are connected and can only be entered or removed as a pair.

Example 12.2 (using a font command) This example uses the typewriter font to simulate on screen LaTeX's verbatim environment. In this environment the output of LaTeX will be literally ("verbatim") the text that is entered and will be set in a monospaced typewriter-like font.

Screen: [**Typewriter:T**1]
Output: [**TEXT**]\verb+**T**1+
Name: Short verbatim+

Screen: [**Typewriter:T**1]
Output: [**TEXT**]\begin{verbatim}**T**1\end{verbatim}
Name: Long verbatim

The first version can be used to make a short verbatim text, for example a LaTeX command. The text cannot contain a "+" because this symbol is used to mark the beginning and end of the verbatim text in the LaTeX output. This restriction can be overcome (at the expense of replacing one restriction by another) by adding other versions where "+" has been suitably replaced.

The second version is used when a large text must be type set just as it is set on screen. Normally, LaTeX will replace multiple spaces by one space and line breaks are placed at optimal places. In the verbatim environment, every space will be used and the line breaks are placed at the positions the user specified.

Math∫pad does not take account of the context in which a symbol is used. Thus, if in one of the versions above, the text place holder, **T**1, contains a special symbol like π, then the symbol π would appear in the screen version whereas the printed output would contain `$\pi$`. This is a trick that can be used by authors of LaTeX manuals to cause the LaTeX code for a construction to appear in the printed document without actually typing it.
□

12.6 Sizes

Just as the use of fonts can enhance readability so can the use of different sizes of symbols. (In mathematics it is common to use smaller symbols for sub and superscripts, of course.) The different sizes can be selected from the pin-up menu that appears when the Size item is selected. The pin-up contains a range of numbers which indicate how large the new font should be relative to the current one in the screen format. Selecting a negative number will decrease the size of the font by a number of steps. A positive number will increase the size.

The use of relative rather than absolute sizes makes it is possible to nest size changes. This is useful when the size of a font should depend on the level of the template. For example, subscripts of subscripted expressions should be yet smaller than plain subscripts.

After a size change the default text font is used This behaviour is based on the way LaTeX (version 2.09) combines size changes and font changes. So, if a small bold font is needed, you have to change the size to small first and then change the font to bold.

The special symbols which indicate the scope of a change of size in the screen format are [**SizeR***n***:** and], the number n being the number that is selected from the pin-up menu. "**R**" is a reminder that the size change is relative (and leaves open the possibility of including an absolute size change in future revisions).

Note that there is a potential discrepancy between the screen format and the automatically generated output since size-change commands in LaTeX (for example, `\large` and `\tiny`) are all absolute and not relative. Special-purpose LaTeX commands, such as the commands for subscripting and superscripting, do accommodate relative size changes but only to a limited extent. (Subsubscripts and subsubsubscripts, for example, are always the same size.) The problem is difficult to eliminate in all situations but in practice hand-editing of the "Output" field in the template specification or adding additional versions usually suffices.

Example 12.3 (specifying a change of font size) A section header is type set in a larger font. In Math∫pad, this can be done with the following template.

Screen: Section [**SizeR2:**[**Bold:T**1]]
Output: [**TEXT**] \section{**T**1}
Name: Section

The template uses a size change in combination with a font change. A size change is used first because it will change the font to the default font as a side effect. The system-generated output field, which would be

[TEXT] Section **T1** ,

has been hand-edited in order to produce the desired effect in LaTeX.
□

12.7 Tabbing

12.7.1 Tabbing and Display Environments

For layout purposes both on the screen and in the output document a robust *tabbing* environment is often necessary. For example, when a number of items are to be listed the margin is moved to the right and a symbol used to indicate the different items as shown below:

- This is an item in a list (of one item!). The margin is positioned to the right of the dot, just before this text.

The tabbing environment is used in a Display expression, available through the main menu. Alternatively, a template (version) can be defined in such a way that a new tabbing environment is created by entering the template in a document. The two menu options Tabbing and Display from the Tabs pin-up menu at the head of the define window are used for this purpose. (The difference between the two is explained below.) Inside a tabbing environment the margin can be moved to the right or left, and tab stops can be set. Math∫pad tabbing environments can be nested to any depth on screen.

The Math∫pad (on-screen) tabbing environment is based on the LaTeX tabbing environment, the former's tabbing commands corresponding one-to-one with those of the latter. The LaTeX tabbing environment is, however, subject to a number of restrictions making its use sometimes undesirable. In many situations there is a special purpose environment available in LaTeX which can be entered in the output by hand. For example, LaTeX tabbing environments cannot be nested but the "enumerate" environment can (so that it is possible to construct numbered lists containing numbered sublists). The Math∫pad tabbing environment should therefore be used for approximating to the LaTeX enumerate environment on screen whereas the appropriate LaTeX code for the enumerate environment should be entered (by hand) in the output section.

Even in situations where the LaTeX tabbing environment, as described in appendix C.9.1 (pages 179–181) of the LaTeX user's guide & reference manual [4], ought to be appropriate it was found to be inadequate to the demands of automatic code generation and therefore a replacement has been developed for use with Math∫pad. The environment is called the *mpdisplay* environment and is defined in the `mathpad.sty` file (to be found in the `latex` subdirectory). The commands that are used are almost the same as those used by the standard tabbing environment. However the mpdisplay environment, has the following four additional arguments:

Elastic space The unit size of "elastic" space (as used also around infix operators).

Tab size The size of the space to be inserted after the current position of the cursor when a move to the next tab stop is to be executed but no tab stop is defined.

Line space Additional vertical space between two lines. This amount of space is added to the normal interline spacing (as defined by the TeX variable `\baselineskip`) within an mpdisplay environment.

Left margin The size of the left margin measured in terms of the number of tab stops.

Math∫pad will insert these arguments when the document is converted to LaTeX and they can be changed via the default window.

The mpdisplay environment is chosen by selecting Display from the pin-up menu under Tabs in the define window. Users not wishing to use the mpdisplay environment may use the LaTeX tabbing environment by selecting instead Tabbing from the same pin-up menu.

12.7.2 Tabbing Commands

The tabbing commands can be selected from the pin-up menu (shown in table 12.5) that appears when the Tabs item at the top of the define window is selected.

A summary of these commands is given below. With the exception of Tabbing and Display, the corresponding LaTeX command is shown in parentheses.

Tabbing

Insert two symbols to mark the opening and closing of the LaTeX tabbing environment. These are [**Tabbing:** and] in the "Screen" field and the LaTeX

Tabs
Tabbing
Display
set
tab
back
plus
minus
push
pop
return

Table 12.5: The Tabs pin-up menu.

code `\begin{tabbing}` and `\end{tabbing}` in the "Output" field. Between these two symbols tabbing commands, such as moving the margin and setting tab stops, can be used. The layout produced by the LaTeX code may differ from that shown on the screen. This is because Math∫pad uses the mpdisplay environment on screen.

Display Insert two symbols to open and close the mpdisplay environment. This is similar to the tabbing environment except for a few commands and the position of the margin. The symbols used are [**Display:** and] on screen and the LaTeX code will be `\begin{mpdisplay}` and `\end{mpdisplay}`. Additional arguments specifying spacing are added when the actual LaTeX file is constructed.

set Set a tab stop at the current position (`\=`).

tab Move to the next tab stop (`\>`). If this tab stop is not defined a space will be inserted and then a tab stop defined. The size of the space is given by the tab size argument of the mpdisplay environment. (See above.) Under the same circumstances the standard LaTeX command would report an error.

back Move to the previous tab stop (`\<`). This command can only be used at the beginning of a line and may give strange results on the screen.

plus Move the left margin one tab stop to the right (`\+`). This command has no effect on the current line and will be ended by the next matching minus command.

minus Move the left margin one tab stop to the left (`\-`). This command has no effect on the current line.

push Save the current positions of the tab stops (`\push`). `\push` is the replacement for the standard LaTeX `\pushtab` command which doesn't work reliably in combination with the plus and minus commands. Several push commands can be nested. push and pop commands must be used in matching pairs within the mpdisplay environment.

pop Restore the positions of the tab stops saved by the last push command (`\pop`).

return Break the line in the LaTeX file without breaking the line in the printed document. When a line in the LaTeX code of a version gets very long and it is to be typeset in the tabbing environment, this return can be used to break the line. The normal newline will insert `\\` when it is used within the tabbing environment.

The LaTeX tabbing commands that are not described here can also be used in the mpdisplay environment although `\mkill` should be used instead of `\kill`. A line in the mpdisplay environment can also be labelled with the `\mplabel` command. Processing the LaTeX file will cause the line to be numbered at the left with an equation number enclosed in parentheses. Each line can have at most one label.

12.7.3 Using the Tabbing Environment

This section contains a number of examples of the use of the tabbing commands. Care should be taken when using these commands because templates are often nested. By keeping to the following guidelines the risk of errors can be kept to a minimum.

- Use the push and pop commands in matching pairs. Every push should be followed by a pop within the same template version. The use of Tabbing or Display in a nested environment is equivalent to a pair of push and pop commands.
- Use the same number of plus and minus commands within a template version. It is not necessary to begin with a plus command because the mpdisplay environment initialises a number of predefined tab positions.
- Move the margin forward to a place holder if the place holder is likely to be filled by something which extends over more than one line. (Thus as a general rule you should always move the margin forward to text place holders unless you have

specific reasons for not doing so.) If this is not done then the layout could be badly compromised.

- Try to avoid using the back command. Usually the same effect can be achieved by using instead a plus command in the line itself together with a minus command in the previous line. The back command can result in strange cursor reactions or strange selections.

- To be sure that the tab stops used in a template will not change the tab stops from other templates, a push and pop command should be added at the beginning and the end of each version with tabbing commands. (This advice is relevant to the LaTeX output only so can be ignored if a special-purpose environment replaces the automatically generated code.)

Example 12.4 (the itemize environment) The LaTeX itemize environment is used to list a number of items, each item being marked by a bullet. In addition, individual items are indented relative to the current text margin. (The text immediately preceding this example is an example of the use of the itemize environment.) Placing a bullet and moving the margin is accomplished in LaTeX by the `\item` command. In Math∫pad this environment can be made with the template

Screen: [**Tabbing:**Item[**set**]ize **T**1]
Output: [**TEXT**]\begin{itemize} **T**1 \end{itemize}
Name: Itemize

to open and close the environment, and the template

Screen: • [**tab**][**plus**]**T**1[**minus**]
Output: [**TEXT**]\item **T**1
Name: Item

for each individual item.

These templates should, of course, only be used in the correct combinations. In particular, the place holder in the first template should only contain a number of instances of the second template. The place holder of the second template can contain everything, including the first template. It is possible in this way to nest any number of these itemize environments within Math∫pad; LaTeX, on the other hand, may not be able to cope if the nesting becomes too deep.

More versions can be added to be able to switch easily between the itemize, enumerate and description environments. One version of the item template should then have an extra place holder in which to enter the argument of the description item.
□

The remaining examples illustrate the use of the tabbing command with elements of a simple programming language called the "Guarded Command Language". In this language repetition of statement S is indicated by writing $\mathsf{do}\ S\ \mathsf{od}$. Conditional terms take the form $b \rightarrow S$ where b is a Boolean expression and S is a statement. No knowledge of the meaning of the terms in the language is needed to follow the discussion since our concern is only about the layout of terms in the language.

Example 12.5 ($\mathsf{do}\ \ S\ \ \mathsf{od}$) The program construction $\mathsf{do}\ \ S\ \ \mathsf{od}$ (where S stands for a statement) can be limited to one line if the statement S is not too long. However, if the statement is long, it is normally set with the od positioned exactly below the do. Further, the statement S begins on the same line as the do; if it is itself a multi-line statement then all its components are indented.

In order to be able to display both long and short do–od statements two versions of a do–od template are created. These are shown below.

Screen: **[Sans Serif:**do] **E**1**[Sans Serif:**od]
Output: **[MATH]$\mathsf{**do**}** **E**1 **\mathsf{**od**}$**
Name: do od (short)

Screen: **[push][Sans Serif:**do] **[set][plus]E**1**[minus]**
[Sans Serif:od]**[pop]**
Output: **[MATH]\push$\mathsf{**do**}~~$\=\+$E**1**$\-**
$\mathsf{**od**}$\pop
Name: do od (long)

The first version is quite simple. The second version is more complicated. The **[push]** and **[pop]** are used to ensure that the tab position defined in the template will not change the tab positions used by other templates. The **[set]** command sets a tab stop a couple of spaces after the do. The margin is then moved forward with the **[plus]** command and is moved back with the **[minus]** command. The combination of these three commands ensures that any multi-line expression will be indented appropriately.
□

The do–od template is used in combination with a guard of the form $\mathbf{E}_1 \rightarrow \mathbf{E}_2$ and a guard separator ($\Box$). The positioning of the separator in the in-line version and in the multi-line version is illustrated by the following example:

$$
\begin{array}{ll}
\mathsf{do} & a \wedge b \ \rightarrow\ \mathsf{do}\ \neg b \ \rightarrow\ a,b \ :=\ b,a \quad \Box \quad a \ \rightarrow\ b \ :=\ \neg b\ \mathsf{od} \\
\Box & a \vee b \ \rightarrow\ b := a \\
\Box & \neg a \vee \neg b \ \rightarrow\ a := b \\
\mathsf{od} &
\end{array}
$$

Example 12.6 (the guard separator) The guard separator is defined to be an infix operator with two versions as shown below.

Screen: $\Box$
Output: **[MATH]**`$\Box$`
Name: Separator (short)

Screen: **[push][minus]**
$\Box$**[tab][plus][pop]**
Output: **[MATH]\push\−**
$\Box$\ > \+\pop
Name: Separator (long)

Again, the first version is straightforward. The second version uses the knowledge that the operator will be used in combination with the second version of the do–od template which moves the margin forward. The operator first pushes the existing tab positions onto a stack and then moves the margin backward. In this way, the box can be placed exactly below the associated do. The **[tab]** command moves the output to the old margin following which the **[plus]** moves the margin forward again in order to ensure that the separator's second argument has the same margin as its first argument.

Moving the margin backwards in this way works in combination with the do–od template but will produce strange effects in circumstances where moving the margin backwards is meaningless.

Note that defining the guard separator as an infix operator allows the possibility of having an arbitrary number of guarded statements in a do–od command.
$\Box$

Example 12.7 (the guard) The guard template should also have different versions, depending on the length of the two expressions. Two possibilities are:

Screen: **[push]E**1 $\rightarrow$ **[set][plus]** **[set][plus]E**2**[minus][minus][pop]**
Output: **[MATH]\push$E**1~~**\rightarrow**~ **$\=\+$**~**$\=\+$E**2**$\-\-\pop**
Name: Guard (short)

Screen: **[push]E**1 $\rightarrow$
[set][plus] **[set][plus]E**2**[minus][minus][pop]**
Output: **[MATH]\push$E**1~~**\rightarrow$**
$**~~~~**$\=\+$**~**$\=\+$E**2**$\-\-\pop
Name: Guard (long)

Both versions set two tab positions: the first one determines where the statement separator (;) should be placed, the second one is used to move the margin to the correct position. The definition of the statement separator would have the same form as the guard separator template discussed earlier.

The first version will move the margin to after the arrow and should only be used if the first expression is not very large. The second version will force the second expression to start on a new line and will indent a little to increase the readability. Another possibility would be to treat the arrow in the same way as the guard separator.

The "Output" tabbing information in the above examples is automatically generated by the system. The " ~ " characters have to be inserted by hand, however, in order to circumvent TEX's rules for eating up spaces. (Each " ~ " in a TEX document produces an interword space. Two other ways of entering spaces are to use "\␣", or `\hspace` with an appropriate length argument, the latter being the most suitable for getting the fine detail right.)
□

12.8 Special Constructions

A page that has been typeset by the TEX system consists of boxes that have been pasted together either vertically or horizontally. The simplest boxes are either character boxes or rule boxes, a "rule" being a rectangle of black ink. Each box has a height, a width and a “baseline”. When pasting boxes together horizontally the baselines are aligned. In a line of text such as this one the baseline is an imaginary line on which each letter sits. Thus, for letters with no descender like “a”, “b” and “c” the baseline is the base of the character; for letters with descender like “p” and “q” the baseline cuts through the character separating the descender from the rest of the character.

Stacking boxes on top of each other to form a new box is a simple way of forming, for example, fractions in mathematics. Suitably defining the baseline of a stack of boxes provides a simple method of aligning their contents with other terms in an expression. Math∫pad has two forms of stacking operation —Stack Base and Stack Centre— which differ in the way that the baseline is defined.

The special symbols used for these and other “special” constructions are available through the pin-up menu (shown in table 12.6) which appears when the Special item is selected at the top of the define window.

Special
Stack Base
Stack Centre
Line
Fill
Dots
Bar
Elastic Space

Table 12.6: The Special pin-up menu.

They are used as follows:

Stack Base Insert a stack with three fields. The baseline of the stacked symbols will be the baseline of the middle field. The Screen-field token is [**StackB: : :**], the colons acting as separators for the three fields. The automatically generated LaTeX will be `\stackrel{ }{\stackrel{ }{ }}`, which is almost never what it should be. Therefore, the LaTeX code of a template which uses a stack should normally be hand-coded by the user.

Stack Centre This stack is almost the same as the stack described above. The only difference is the position of the baseline, which is a half `ex` below the baseline of the middle field. The Screen-field token is [**StackC: : :**].

Line Fill the (remainder of the) stack field with a horizontal line. The length of the line will depend on the width of the other fields. It can be used for boxes, fractions, underlining or overlining. The Screen-field token is [**Line**] and the automatically generated LaTeX code is `\hrulefill`.

Fill Fill the (remainder of the) stack field with white space. Just like the line, the length of the space will depend on the width of the other fields. Such a filler is used to align the different fields at the left or right. To align to the left fill on the right, and to align to the right fill to the left. [**Fill**] is inserted in the Screen field and `\hfill` in the Output field, respectively.

Dots Fill the (remainder of the) stack field with a dotted line. The length of this horizontal line will depend on the width of the other fields. The token used in the screen field is [**Dots**]; the corresponding LaTeX code is `\dotfill`.

Bar Insert a variable length vertical line. Within a stack the length of the line will be the height of the field in which it is used. When the bar is used outside of a stack the height of the line will be used. The Screen-field token is [**Bar**]; the corresponding LaTeX code is the "|" symbol.

Elastic Space Insert elastic space. The size of this space will stretch as the size of the terms filling the place holders in an instance of the template grows. The unit space is specified in the default window and is the same as the unit space around infix operators. The token used in the Screen field is [**Space**]; the corresponding LaTeX code is the macro `\ms` (defined in `mathpad.sty`) together with a parameter giving the number of units of elastic space to be inserted.

Do not confuse elastic space with filler space. Filler space is used only in stack constructs, elastic space may be used anywhere, and the rules governing their size are quite different. Note also that bars are most often used in stack constructions but they may also be used elsewhere.

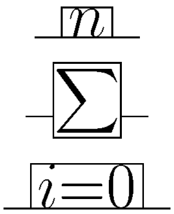
12.8.1 Stacking Expressions

The two screen format stacking commands, Stack Base and Stack Centre, are used to simulate on screen the various LaTeX commands for putting things above and below each other. Both of these commands have three arguments the contents of which are to be stacked one above each other in a vertical column.

Each argument defines a box with a *width*, *height*, *baseline* and *descender*. For example the left side of the expression

$$\sum_{i=0}^{n} i$$

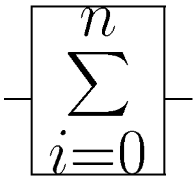
is built with the following boxes:



The box parameters are determined as follows:.

- The *width* of the box is the width of the text inside the box.
- The *height* of the box is the distance from the baseline to the top of the box. The *baseline* of the box is indicated in the diagrams above by the lines protruding from the left and right sides. The height is negative if the box is entirely below the baseline.
- The *descender* of the box is that part of the box from the baseline to the bottom of the box. The vertical size of the descender is called the *descent*. The descent is negative if the box is entirely above the baseline.

With the stack construction these three boxes can be put on top of each other. The boxes will be centred unless specified otherwise. The three boxes form a new box with a new baseline:



A number of LaTeX constructions have a way to specify where the baseline should be placed This is usually done with an optional argument such as `[t]`, `[b]` or `[c]`. In Math∫pad, the position of the baseline can also be changed. However it can only be set to one of two positions:

- at the same position as the baseline of the middle box. (Stack Base)
- a half ex below the baseline of the middle box where ex is the height of an x. (Stack Centre)

The latter option makes it possible to create subscripts, superscripts and fractions as well as many other constructions.

The length of a variable line or space will be the difference between the length of the longest field and the field in which the line or space is used. If multiple lines or spaces are used inside one field, the available length will be divided across the different lines and spaces.

Math∫pad centres the three fields of a stack, which is equivalent to adding a space at the left and right of every field. If a field is to be left-aligned, a space should be added at the right side of the field. A right-aligned field can be made by adding a space at the left side of the field.

Making a selection in a centred or right-aligned text may be a little difficult. The cursor can appear at the wrong position. This will be solved as soon as possible.

Some examples using the Stack constructions

Many difficult constructions of LaTeX can be simulated with the two stacking commands. Some examples are described here to illustrate their use. Each example starts with a template definition followed by an explanation of how to use it and what variations are possible.

Example 12.8 (subscript)

Screen: **E**1[**SizeR-1:**[**StackC: ::E**2]]
Output: [**MATH**]$**E**1_{**E**2}$
Name: Subscript

This template can be used in math mode and the [**SizeR-1:**] code is used to make the size of the subscript one size smaller. In this way, it is possible to add a subscript to a subscript.

If the first expression is removed, the subscript can be used as a postfix operator. In this case {} should be added before the _ in the LaTeX code.

The space in the first field of the Stack Centre command has been entered in order that the template can be selected with a single click. Clicking above the subscript will select the template, not the subscript. If the space is omitted, clicking above the subscript will select the subscript. The template can still be selected by dragging over the subscript and the expression.
□

Example 12.9 (summation symbol)

Screen: **[StackB:[SizeR-1:E**2**]:[SizeR4:**$\sum$**]:[SizeR-1:E**1**]] E**3
Output: **[MATH]$**\displaystyle{**sum**_{**E**1}^ {**E**2}}\ **E**3**$**
Name: Displayed sum

Screen: $\sum$**[StackC:[SizeR-1:E**2**[Fill]]: :[SizeR-1:E**1**[Fill]]] E**3
Output: **[MATH]$**\textstyle{**sum**_{**E**1}^ {**E**2}}\ **E**3**$**
Name: Inline sum

These two versions of the sum template can be used to make $\sum\limits_{\mathbf{E}}^{\mathbf{E}} \mathbf{E}$ and $\sum_{\mathbf{E}}^{\mathbf{E}} \mathbf{E}$. The first version uses a very large summation symbol and places two smaller arguments above and below it. The three fields of the stack are centred. The LaTeX code contains the `\displaystyle` macro which is used to force LaTeX to place the two arguments above and below the sign instead of to the right. With Math∫pad it is possible to use a displayed summation as one of the arguments of another summation. The size of this summation will be smaller in Math∫pad but LaTeX provides no macros to do this.

The second version places the two arguments at the right side of the summation symbol. The two arguments are placed above each other and are left aligned by adding the **[Fill]** at the right side. The `\textstyle` macro in the LaTeX code has the same function as the style macro in the first version.

The size of the arguments will decrease when the templates are nested. Math∫pad does not check if LaTeX will use the same size. Especially with subscripts, superscripts and constructions like the summation, LaTeX has a limited depth of nesting with respect to the size of the argument.
□

Example 12.10 (underlining)

Screen: [**StackB::T**1:[**Line**]]
Output: [**TEXT**]\underline{**T**1}
Name: Underline text

Screen: [**StackB::E**2:[**Line**]]
Output: [**MATH**]$\underline{**E**2}$
Name: Underline expression

Screen: [**StackB:**[**Line**]:**E**2:]
Output: [**MATH**]$\overline{**E**2}$
Name: Overline

Screen: [**StackB::T**1**E**2:[**Line**]]
Output: [**TEXT**]\underline{**T**1$**E**2$}
Name: Underline both

The underline and overline template shows how the variable line can be used. The different versions make it possible to underline text and to overline expressions. The last version is needed to install the template: every template must have one version which contains all place holders.

The text that is underlined should not be too long. Math∫pad will only underline the last line of the text. LaTeX will not break the text at all.

The position of the line depends on the largest descender, just like in LaTeX. In some situations this is not what you want: compare the complete underline and the separated underline in this sentence. The separated underlining looks nicer but more difficult to make with LaTeX. With Math∫pad, just select the parts and use the underline template.
□

Example 12.11 (`\fbox` **command)**

Screen: [**StackB:**[**Line**]:[**Bar**] **T**1 [**Bar**]:[**Line**]]
Output: [**TEXT**]\fbox{**T**1}
Name: Box

The box template adds a box around text. The text should not contain multiple lines although the stack can be used in the text. The metrics of the box will match the metrics of the text.

The two lines are used to over- and underline the text at the same time. The two bars will add the vertical lines of the box. They are placed inside the stack because the length of the bar should be the height of the text, not the height of the line that contains the box.

Just like the underline, the box disables the line breaking routines of LaTeX. So, the contents of a box should not be too large to ensure a correct LaTeX output and a correct layout by Math∫pad.

The template could be improved by adding the minipage and tabbing environments of LaTeX. The layout of LaTeX will then be correct in almost every case. Math∫pad will still have some trouble displaying it correctly.

The box will leave a little space between the text and the edges. This space can be removed by adding the code `\fboxsep 0pt` before the `\fbox` command
□

Example 12.12 (fractions)

Screen: [**StackC:E**1:[**Line**]:**E**2]
Output: [**MATH**]$\displaystyle{\frac{**E**1}{**E**2}}$
Name: Displayed fraction

Screen: [**MATH**][**SizeR-2:**[**StackC:E**1:[**Line**]:**E**2]]
Output: $\textstyle{\frac{**E**1}{**E**2}}$
Name: Inline fraction

Fractions can be entered in both displayed and inline versions. Nesting of fractions also poses no problems. A difficult fraction like

$$\frac{\ln(n+1)}{\ln n} = \frac{\ln n + \ln(1+\frac{1}{n})}{\ln n} = 1 + \frac{\ln(1+\frac{1}{n})}{\ln n}$$

is easy to make with Math∫pad. As in the example, inline and displayed fractions can be mixed in an expression; also, changing between the two versions is very simple. Of course, LaTeX has to support the construction you have made but it will never be much different from the expression that Math∫pad displays.

In some situations the LaTeX code produced is not optimal. The style commands are not always necessary but they ensure that the layout of LaTeX will resemble the layout of Math∫pad.
□

Example 12.13 (roots)

Screen: **[StackB: [Line]:**$\surd$ **E**1]
Output: **[MATH]$**\sqrt{**E**1}**$**
Name: Root

Screen: **[StackB:[SizeR-2:E**2]**:[Fill]**$\surd$**:][StackB:[Line]:E**1**:]**
Output: **[MATH]$**\sqrt[**E**2]{**E**1}**$**
Name: n-th root

The root template uses the line as a part of the root symbol. LaTeX produces a much better root symbol but you can see directly what is meant by

$$\sqrt{\frac{x}{y}}=\frac{\sqrt{x}}{\sqrt{y}} \quad \wedge \quad \sqrt[4]{x}=\sqrt{\sqrt{x}}$$

although it does not look completely WYSIWYG on the screen.
□

Example 12.14 (vectors)

Screen: **[StackB:**$\rightarrow$**:E**1**:]**
Output: **$**\vec{**E**1}**$**
Name: Vector

Screen: **[StackB:[line]**$\rightarrow$**:E**1**:]**
Output: **[MATH]$**\overrightarrow{**E**1}**$**
Name: Long vector

This template can be used to add an accent, in this case a vector. Every possible symbol can be used for this purpose because the stackrel and shortstack macros can be used to make a correct LaTeX code.

The second version uses the fact that a variable line will be placed centred with respect to the height. The line of the arrow is positioned in the middle, so the line will enlarge the arrow.
□

More on Bar

A bar may be used on its own (thus not as part of a stack). On screen the bar will have the same height as the template in which it appears. For example, one can create a template with one text place holder of relative size +5 and parenthesised by bars. An instance of such a template is shown below.

$$\left|\text{xyz}\right|$$

The LaTeX that is automatically generated in such circumstances may *not* be correct because in LaTeX (and TeX) a bar often has a special meaning and its use in text is unreliable.

12.8.2 Elastic Space

Elastic space is space in a template that grows with the complexity of the expressions occupying the template's place holders. The use of elastic space is usually, but not necessarily, confined to mathematical structures. It is automatically added around infix, prefix and postfix operators; for multifix operators its inclusion can be specified with the Elastic Space menu item. As an example, suppose a notation is required for universal quantification of the form:

$$\forall(\mathbf{V}{:}\mathbf{E}{:}\mathbf{E}) \quad .$$

The notation has a variable place holder and two expression place holders, the first for the domain of the quantification and the second for the term of the quantification as in, for example,

$$\forall(x\colon\ x{\in}S\colon\ p.x) \quad .$$

The parentheses and colons serve as delimiters.

For such a notation it may be desirable to increase the spacing around the variables and the two expressions as their complexity increases. This can be a significant help to a reader in recognizing the structure of the expression. In the version specification below the two occurrences of "**[Space]**" specify that elastic space is to be inserted after each colon in the template. They are entered simply by choosing the Elastic Space menu item in the define window.

Screen: ∀(**V**1:**[Space]E**3:**[Space]E**2)
Output: **[MATH]$\forall(V**1{:}**\ms{0}E**3{:}**\ms{0}E**2)**$**
Name: for all

As an example of the effect, compare the spacing around different subexpressions in the following expression.

$$\forall(x{:}\forall(y{:}p{:}q){:}\forall(y{:}p{:}q))$$

Calls to the `ms` macro are automatically removed in the output generated by any instance of the template for which the value of the parameter is zero (i.e. no elastic space is to be inserted). Note also that the braces around the colons in the Output section are necessary to prevent LaTeX from imposing its own spacing conventions. These are not automatically inserted.

12.9 Hints on Testing

- A click on the Update button in the define window causes the template in the stencil to be overwritten by the contents of the template specification windowpane. Thus, when experimenting with the definition of a template, it may be advisable to click on New to create a new template for testing purposes. When you are satisfied that the new definition is as you wish it to be, the following procedure can be used to update the original template.

  Firstly ensure that all files that use the existing template are loaded. Then make the old template identical to the new one by clearing the old definition and copying across the new definition. Install using the Update button. This will have the effect of modifying accordingly all occurrences of the template in the loaded documents. Your stencil will, however, contain two templates that are identical but for their code numbers. Remove one of them by loading its definition into the define window and clicking on Double whilst holding the control key down.

- The "Screen" field definition can be checked within the define window. The keyboard command **M-T** causes the system to preview the actual screen format (thus the format of each version of the template as it will actually appear on the screen). Typing **M-T** again causes the system to toggle back to the command language format.

- Output and Paste under the Misc.button on the console can be used in combination to check the LaTeX code produced by the template. Select with the left mouse button an instance of the template in a document, click on Output, move the cursor to a suitable position (possibly in a scratch document or the buffer) and click on Paste.

12.10 Keyboard Shortcuts

The most commonly used commands can also be entered from the keyboard using the key sequences shown in table 12.7.

Key	Token	Key	Token
Tab	**[tab]**	**M-=**	**[Stack?:::]**
S-Tab	**[back]**	**CM-E**	**E**n
C-Tab	**[set]**	**CM-I**	**I**n
C-Return	**[NwLn]**	**CM-O**	**O**n
M- $-$	**[Line]**	**CM-T**	**T**n
M-space	**[Fill]**	**CM-V**	**V**n
M-.	**[Dots]**		

Table 12.7: Keyboard shortcuts for version definition.

In the case of a stack (**M-=**) a prompt for the type of stack appears in the console. The letter "b" for baseline stack or "c" for centred stack should be entered. The system will beep if neither of these is entered.)

Chapter 13

Defaults

13.1 The Default Window

The default window, as shown in figure 13.1, can be opened by selecting the Default option of the Window menu. The values specified in the default window are used to determine spacing both on screen and in the output produced by LaTeX, as well as the paths for the different file types and the fonts that are used.

The three buttons labelled Set, Save and Done at the top of the default window are used as follows:

Set Update the default values to those specified in the default window. The layout of any loaded documents will also be updated, if necessary. Any numerical values are checked to make sure that they are reasonable.

Save Update the default values and save them in a file named "`.mpdefaults`". This file is loaded whenever the editor is started without a project being specified.

Done Close the default window ignoring any changes since the last use of the Set or Save button.

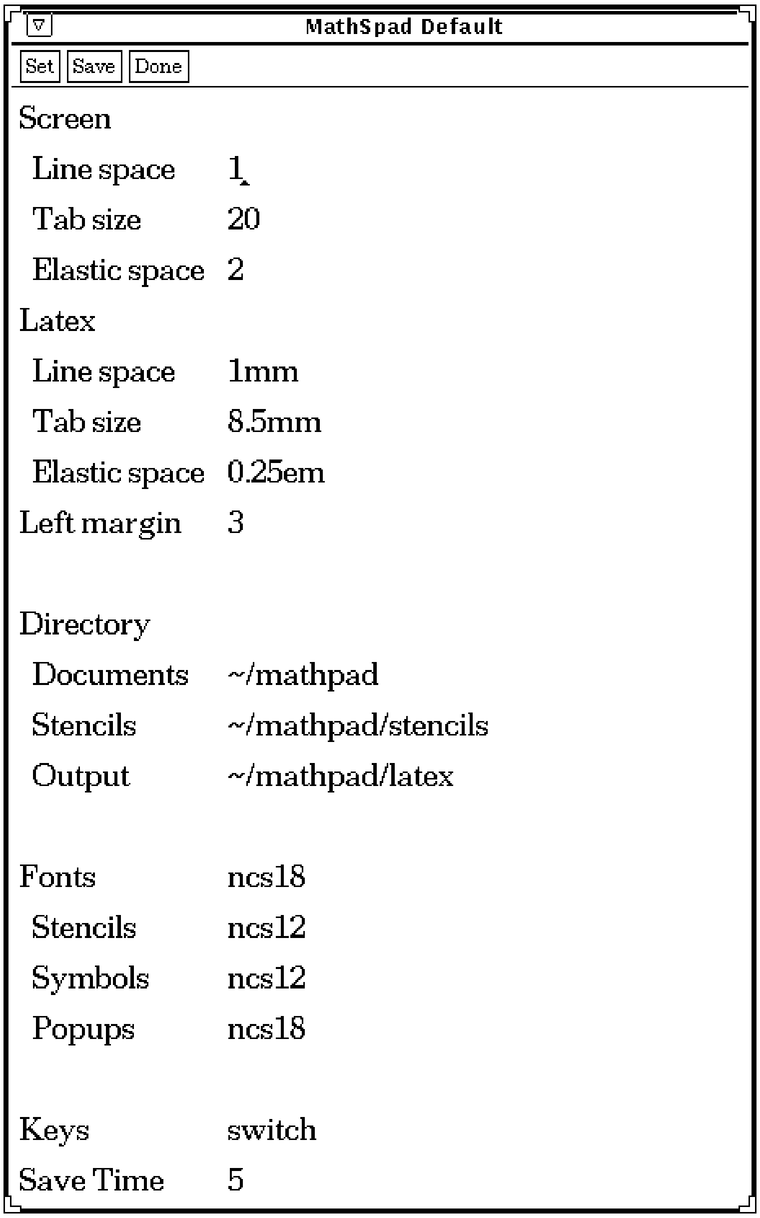


Figure 13.1: The Default Window

13.2 The Settings

In the windowpane a number of fields are displayed. At the left side of the windowpane a description of each field is given, at the right side the default value is specified. The different fields are:

Screen These fields define the layout on the screen. The three values can be adjusted so that the screen layout corresponds with the layout that LaTeX produces. The values are numerical and the unit is a screen pixel.

- **Line space** The additional vertical space between two lines.
- **Tab size** The default distance between two tab-positions.
- **Elastic space** The length of one unit of elastic space inserted around operators and where specified in templates.

Latex These fields define the layout that LaTeX will produce. Each field contains a string which is copied into the LaTeX code and should define a length, for example 12.7mm or 0.5in. Note that something like `\parindent` is also a correct length definition. (Consult your LaTeX manual for details of what is a valid length definition.) These strings are not checked so the user should be careful. The meaning of the fields is the same as the meaning for the screen fields.

Left margin This value defines how many tab positions there should be between a displayed expression and the left margin. The size of a tab is defined by the "tab size" field.

Directory There are 3 directories the user can change. These directories are used to store the different files or to search for files. The different files will be allocated to the directories in the following way:

- **Documents** Documents (i.e. `.mpd` files).
- **Stencils** Stencils (i.e. `.mps` files).
- **Output** Output (usually LaTeX, i.e. `.tex`) files.

Fonts This name is used to select a group of fonts. When Math∫pad is started the file with that name will be used to load the different fonts. The specified group of fonts is used in edit windows, the buffer window, the find & replace window and the lower part of the define window. The fonts for the stencil windows, the symbol windows and the pop-up menus can be specified separately:

Stencils The fonts for the stencil windows.

Symbols The fonts for the symbol windows.

Buttons The fonts for buttons, the default window and the upper part of the define window.

If one of these fields is empty, the font for documents will be used for the related windows. By clicking on the description of a field, a menu with the available groups of fonts will appear from which one can be selected.

Keys The available keyboard commands can be changed by specifying a number of keyboard files. These files must have the extension "`mpk`" (The extension will be added automatically so do not enter it.) Multiple filenames are separated by colons (:). See chapter 14 for further details.

Save Time The time between two periodic save operations can be specified here. A period will start when an edit action is performed and the save operation will only save those documents and stencils that have been changed since the last save operation. The saved documents and stencils get the name *#name#*. Documents and stencils will also be saved in a dump file if Math∫pad has to stop for some reason. The specified time can be between 1 and 60 minutes.

13.3 Editing Commands

The different fields are connected so the cursor can be moved from one field to another. The cursor can also be moved with the mouse by clicking on the new position or by dragging inside a field.

When the string inside a field is too large it will be truncated at the edges, possibly in the middle of a character. By moving the cursor or resizing the window, the entire string can be viewed.

In numerical fields only digits can be entered while in a text field all editing characters can be entered. In addition to editing commands discussed in chapter 6, **C-return** and **C-tab** can be used to move to the previous field and **return** and **tab** can be used to move to the next field.

Chapter 14

System Administration

One of the most useful aspects of computer software is that it *can* be customised to one's own requirements. The time that is spent doing so often pays for itself handsomely within a very short space of time. One of the most frustrating aspects of computer software is that it often *must* be customised so as not to conflict with the local configuration before one can begin to use it. Two aspects of Math∫pad that you may want to, or may have to, reconfigure before you can use the system to full advantage are the keyboard commands and the installed fonts. How to do so is described in this chapter.

14.1 Customising the Keyboard

By default, Math∫pad provides a keyboard binding resembling the Emacs keyboard binding. Customising the keyboard may be necessary because the window manager or the operating system you are using preempts some of the keyboard bindings for its own use. (For example, on the Linux system used by one of the authors the window manager preempts the **F7** key in order to iconicize windows and so the key can't be used for copying expressions.) You may also want to customise the keyboard if, for example, you are used to another text editor with different settings to Emacs or you commonly write in a language other than English and want to be able to enter particular accented letters simply and efficiently.

To customise the keyboard requires adjusting the keyboard definition files. These have the extension mpk. For details of the Emacs keyboard bindings see the file emacs.mpk supplied with the distribution. (We recommend that you consult this file in any case as an illustration of the topics discussed in this section. If you don't know where it is located consult your system manager or the one who installed the system for you.) Once complete, the name of the keyboard definition file (or files) that you want to use should be entered in the "Keys" field in the Default window. (See chapter 13.)

An mpk file is a text file containing comments and up to five different kinds of definition. A comment line should start with a hash ("#"). Math∫pad will collect non-comment lines until it finds a line ending with a semi-colon (";"). The collection of lines will be parsed according to the format

(*definition type* : *argument* [: *argument*]);

where *definition type* can be one of:

def-key Add a function to a sequence of keys in a particular key map.

def-key-range Add a function to a range of keys in a particular key map.

def-map Define a new key map.

redef Redefine a key to simulate another key.

def-func Define a new function. (This option is currently rather limited).

Each definition type has its own set of arguments. An argument can be one of:

string A string between quotes ("""). A string can be an argument of a function, the name of a key map or the name of a new function.

key The name of a key between brackets ("[" and "]"). To find out the name of a key depress the key after the key combination **C-** \ .

key sequence A sequence of keys.

function A function call. The format of a function can be either *name*(*key*,*string*) or *name*(*key*,*number*), where *key* is optional. The name of the function can be either the name of a predefined function or the name of a user-defined function. Since user-defined functions don't use their arguments in the current implementation, any argument will be ignored.

14.1.1 Adding a Function to a Key Sequence.

Use the def-key definition type to add a function to a key sequence. The format of the definition is

def-key : *map name* : *key sequence* : *function*

where *map name* is the name of an available key map, *key sequence* the sequence of keys leading to the function and *function* the function to call. There are a number of predefined key maps as shown in table 14.1.

Name	Purpose
Global	The document windows, the buffer and the find & replace window.
search	Incremental search commands.
answer	Answers to incremental search queries.
Define	The define window.
String	The input fields in the default window, the file selector and the pop-ups.

Table 14.1: Predefined key maps.

The def-map type is used to add new key maps to this list. There are also functions to switch to a different map.

Key sequences are used to create prefix commands. For example, if you want to insert accented characters with the prefix command **C-V**, you can use definitions like:

```
(def-key:"Global": [C-V][e][`]: insert-symbol(232));
(def-key:"Global": [C-V][e][^]: insert-symbol(234));
```

If you now type **C-V e `**, Math∫pad will insert symbol number 232, which is character è. Typing **C-V e ^** , will give you symbol number 234 (ê). If you make a mistake somewhere in between and type something like **C-V e 6**, Math∫pad will separate the list of keys into valid key combinations until the list is empty or a valid prefix remains.

It is also possible to bind templates to key sequences. The following commands bind key sequences to the first and second versions of the Accent template (in the **latex2e** stencil). The stencil should be loaded in order to use the key sequences.

```
(def-key:"Global":  [multikey][a][']: insert-template("Accent:1"));
(def-key:"Global":  [multikey][a][`]: insert-template("Accent:2"));
```

14.1.2 Adding a Function to a Collection of Keys.

The def-key-range definition type adds a function to a collection of keys in a certain key map. The format of the definition is

def-key-range : *map name* : *key1* : *key2* : *function*

where *map name* is the name of a key map, *key1* the first key of the range, *key2* the last key of the range and *function* the function to call.

The effect is to assign the function to all the keys in the given range, the entered key being passed as an optional argument to the function call. For instance, the definition

```
(def-key-range:"Global":[ ]:[~]: self-insert(1));
```

adds the self-insert function to all the keys on a conventional QWERTY keyboard. The self-insert function will use the entered key as an argument and will insert that key as a character at the cursor position. The argument "1" is the number of times the character is to be inserted.

14.1.3 Defining a New Key Map.

Use the def-map definition to define a new key map. The format of the definition is

def-map : *map name* : *key sequence* : *new name*

where *map name* is the name of an existing key map, *key sequence* the sequence of keys leading to the new key map and *new name* the name of the new key map.

New key maps are used to create different input modes. After you have defined a new key map, you can fill the map with assignments of key sequences to functions, just as in the default key maps. For instance, if you want to define a shortcut for accented characters, you can define a map `Accent`, and fill it with self-insert functions:

```
(def-map: "Global": [C-Q]: "Accent");
(def-key-range: "Accent": [C-A] : [C-?]: self-insert(1));
```

The range **C-A** up to and including **C-?** is defined in accordance with the ISO-5589-1 standard, which means that you can use the control and meta keys in combination with the normal keys to get accented characters.

14.1.4 Changing Keys.

Use redef to re-map one key to another key. The format of the definition is

redef : *key1* : *key2*

where *key1* is the key you want to re-map and *key2* is the new meaning of the key. For instance, if your keyboard doesn't have a help key, you can define the key **C-H** to be that key with the definition line:

```
(redef : [C-H] : [help]);
```

The function previously associated with the **C-H** will now be unavailable, unless you redefine a different key to have that function.

14.2 Font Management

A font or typeface is a design of a collection of related symbols. The letters of the alphabet that you are reading now are all drawn from a particular font (the name of which is not relevant). Sans serif letters form a different font. Fonts can also contain designs that one desn't normally "read" — pictures of chess symbols, and heraldic symbols are examples of designs that might make up a font. The elements of a font are often called *glyphs*.

Scientific documents usually make use of a lot of different fonts, rather more than their fair share in fact. The advantage of using a system like Math∫pad is that a great many fonts are already available for use both on-screen and in the printed output, and there is no limit to the number you can add. Math∫pad will only load those fonts that are actually used in a document and if a font can't be loaded, a default will be substituted.

The design of new fonts is a task for experts. If you are not an expert, there are lots of (electronically available) fonts to choose from. Incorporating one in Math∫pad is complicated, at least initially, because of the many ways that fonts may be represented. After the first attempt, however, it will become routine and you will discover for yourself the fun and usefulness of using the facility for all sorts of purposes such as editing chessboards or including fancy headers.

To find out what is available on your local system use the program `xlsfonts`. This will generate a (usually long) list of available fonts. Alternatively, the command `xfd -fn 'fontname'` can be used to check if a particular font is available.

There are two aspects to installing a new font. For each glyph in the font you have to specify what you want to appear on the screen and what you want to be entered in an output file. You also have to say if you want the font to be entered in the book of symbols available through the symbol window and how the font should be scaled (for example when glyphs of the font are used as subscripts).. The hardest part is saying what you want to appear on the screen because it involves instructing the X window system to take notice of the new font. This aspect of the task is discussed in section 14.2.1. For the remaining tasks it is necessary to insert an entry in the `fonts.mpt` file for each new font to be added and create three files for each new font — the `mpt` file, the `mpm` file and the `mpf` file. These tasks are detailed in sections 14.2.2, 14.2.3, 14.2.4 and 14.2.5.

14.2.1 Making Fonts Available to X

If you have a font that you want to use in Math∫pad, you need to find a way to add that font to the font database of the X Window System. This is usually not that difficult.

First you have to convert your font to a format which is readable by the X Window System. So, you may have to use a conversion tool to get a bdf, pcf or Type 1 font. Bdf and pcf fonts are bitmap fonts, which means that they do not scale well. Sometimes the X Window System will try to scale bitmap fonts, but this will usually look awful. A Type 1 font is an outline font and the X Window System can scale it. So, if you have an outline font, try to convert it to a Type 1 font or create a collection of bdf fonts of different sizes. If you have a bitmap font, convert it to a bdf font and then scale and edit it to your own wishes.

Table 14.2 contains a short list of available font conversion programs. If you can't find a program suitable for your problem, you could try looking at the frequently asked questions of the comp.fonts newsgroup.

Tool	Purpose
bdftopcf,dxfc	From BDF fonts to PCF fonts. Using PCF fonts improves the performance of the X server.
bdftosnf,xtbdftosnf	From BDF fonts to SNF fonts. SNF fonts are used by old X servers.
convertfont	From BDF fonts to X11/NeWS fonts. Sun OpenWindows uses this format.
mac2bdf,mac2win	From Mac Bitmap fonts to BDF fonts.
pktobdf	From TEX's pk fonts to BDF fonts.
gftopk	From TEX's gf fonts to TEX's pk fonts.
mf	From METAFONT outline fonts to a TEX's gf fonts

Table 14.2: Font conversion programs.

Once you have a font in the correct format, you have to make the X server aware of the extra font. The X server uses a path of directories to search for fonts. Each directory needs to have an index file called `fonts.dir`, which is created with the command `mkfontdir`. This program uses the file `fonts.scale` to find the names of the Type1 fonts and checks the available fonts in the directory for extra font names. So, if you want to add a Type1 font, you first need to update or create the file `fonts.scale`, so that it contains a line with the file name and the font name. You can use the file `fonts/Type1/fonts.scale` from the X11 distribution as a guide.

After the index file is updated, you have to tell the X server that it should read the new index file. If the X server already searches the directory for fonts, the command `xset fp rehash` will have the desired effect; otherwise you have to use the command `xset fp+` <directory>. You can test if everything has worked with the commands `xfontsel` and `xfd`.

Known Problems. Sometimes it is a little difficult to add a directory to the font path of the X server. If you have problems making the X server aware of your font directory, check out this list of solutions:

I'm using OpenWindows and nothing seems to work.

OpenWindows or XNeWS uses its own font format and different program to convert them or to make the index files. You have to use `convertfont` and `bldfamily` instead of `bdftopcf` and `mkfontdir`. It creates .fb, .ff and Families.list files.

mkfontdir doesn't work.
On some systems, vendors have created their own versions of `mkfontdir`. Other possible names are `dxmkfontdir` and `xtmkfontdir`.

The X server crashes when I try to load the font.
On some systems, the X server crashes if the index file is not created correctly. You have to create a new index file to solve it.

I did a remote login and I can't add my font directory.
The X server can only use font directories which are available on the host on which the X server is running. You might have to copy your fonts to a directory which is available to the X server or you may have to set up your own font server (see the next item).

I'm working on an Xterminal.
Xterminals often use a font server to provide the fonts and you can't add font directories to the font path. To get around this, you can set up your own font server. A program called `xfs` can be used for this and it isn't very difficult. There is an example in the font directory of the distribution.

I'm using an X emulator.
An X emulator is used to run X programs on a system which can't handle these programs itself. If you want to use X fonts in that environment, you usually have to convert the fonts with a tool which is part of the X emulator package.

14.2.2 General Font Information

Math∫pad uses the file `fonts.mpt` to specify additional information about fonts. (This file is stored in the Math∫pad library directory.) For each font the following items should be specified:

Fontnr.: The internal number of the font. The lower numbers are used for standard fonts. So, if you add fonts, choose a higher number (100 to 219). Font number 0 is the default text font. The highest numbers (220 to 255) are reserved for internal representations.

Fontname: The name of the font. This name will be used in the pop-up that appears in the define window (under Fonts) or when the font of an identifier is changed.

Latex: The name of the `mpt` file defining the output macros for this font. (For most uses of the system the "output" macros will be LaTeX macros. If you are using the system to generate code in some other mark-up language don't be put off by the heading.) See the next section for how to construct the `mpt` file.

In addition, the following two items should be included if necessary.

Openfont: The string used to open an environment in which the font is to be used. (The string used in LaTeX to open the italic environment, for example, is "`\mathit{`".)

Closefont: The string used to close the environment. (This is always "`}`" in LaTeX.)

As a running example we consider adding a "chess" font. To do so, first add the following to the `fonts.mpt` file.

```
Fontnr: 53
Fontname: Chess
Latex: chess.mpt
Openfont: {\chessfont
Closefont: }
```

14.2.3 Attaching Macros to Glyphs

To generate valid output, Math∫pad has to know which macro should be used for each glyph. So, for every font you use, you have to create a file which contains a macro name for every glyph in the font you want to use. These files have the extension `mpt` and are plain ascii files. You define a macro in one of the following ways:

```
<nr>&  <macro-both>
<nr>$  <macro-math>
<nr>   <macro-text>
```

In this example, `<nr>` is the number under which the glyph is known by the X window system and `<macro...>` is the macro you want to associate with that symbol. The macro will start with the first visible character and end with the last character on the same line. The three ways to define a macro are based on the modes that LaTeX uses. The macro can be used in math mode, text mode or both and Math∫pad has to know which macro should be used in which mode. To make sure that a macro does not mess up the output, Math∫pad will check if the LaTeX output is valid and will add a space or a `{}` pair in order to end a LaTeX macro correctly.

If your font uses some uniform encoding, you can probably adapt or copy one of the available `mpt` files. If you have to create the `mpt` file from scratch, you should use a program like `xfd` to display the font while you are editing the `mpt` file. That way, you can see the symbols and check which number each symbol has, so you don't make mistakes. If there is a file which somehow defines macros for the symbols, like a LaTeX style file, you might want to consider to convert that file automatically.

14.2.4 Switching Between Sizes

In Math∫pad you can switch between different sizes of fonts if you have multiple versions of that font. How Math∫pad should switch from one size to another is specified by an `mpm` file. The file should contain something like this:

```
8: *,10,12,14,16,18,20,24 : -*-chess-*-*-*-*-8-*-*-*-*-*-*-*
10: 8,*,12,14,16,18,20,24 : -*-chess-*-*-*-*-10-*-*-*-*-*-*-*
12: 8,10,*,14,16,18,20,24 : -*-chess-*-*-*-*-12-*-*-*-*-*-*-*
14: 8,10,12,*,16,18,20,24 : -*-chess-*-*-*-*-14-*-*-*-*-*-*-*
16: 8,10,12,14,*,18,20,24 : -*-chess-*-*-*-*-16-*-*-*-*-*-*-*
18: 8,10,12,14,16,*,20,24 : -*-chess-*-*-*-*-18-*-*-*-*-*-*-*
20: 8,10,12,14,16,18,*,24 : -*-chess-*-*-*-*-20-*-*-*-*-*-*-*
24: 8,10,12,14,16,18,20,* : -*-chess-*-*-*-*-24-*-*-*-*-*-*-*
```

Every line contains a number, a list of numbers describing how to switch between versions and the name of a font. The number is used as a reference number and is usually related to the size of the font, although it may be any number. The list of reference numbers indicates how Math∫pad should switch from one size to another. A star in that list indicates the relative position of the font in size order. It is possible to use a reference number more than once, which is useful if you only have a restricted set of fonts. In such a situation, Math∫pad will use the same font for different sizes. The name of the font can be any name the X server accepts.

14.2.5 Font Selection

After creating the `mpm` file, you can add the font to a font selection file having the extension `mpf`. The font selection file describes which fonts are used at which default size. For each font you have to specify the following items:

Font group: The default font you want to use. You specify the font with a reference number and the name of an `mpm` file. The default font will be the font that is associated to the reference number in the given `mpm` file.

Number: The internal number of the font.

Page: The pages you want to see in the symbol palette. The numbers you need to specify are the numbers of the symbols for the upper left corner of a symbol page. You can specify at most two symbol numbers or 'none' if you don't want any pages for this font to appear in the symbol palette. If you don't specify this item, Math∫pad will show all the symbols of that font on one or two pages, depending on the number of symbols in that font.

For example,

```
Fontgroup: 12, chess
Number: 53
Page: 32
```

will add the font described in `chess.mpm` to the available fonts. The default size will be 12 and Math∫pad will use font number 53 for its internal representation. If this font selection file is used for the symbol palette, a symbol page will be used to display the (at most) 128 symbols starting with symbol number 32.

Appendix A

Advice for Newcomers to LaTeX

by Hilary Backhouse

Once you have completed writing a document using Mathpad you will most certainly want to print the document on paper (get a "hard copy" in computer jargon). Mathpad does not do this for you; instead, it can be made to create a file written in a mark-up language, which file is then processed by a computer program to produce the desired output.

Since the output of a Mathpad document is processed by a computer program the format and content of the output file must obey rigid restrictions. Consequently, in order to use Mathpad successfully in this way, some knowledge of the mark-up language and "printer" program is essential.

In this chapter we give advice on preparing documents with Mathpad for subsequent processing by LaTeX. It is a basic guide and is primarily intended for users with no experience of LaTeX. Its aim is to enable such users to produce satisfactory output using existing templates. Users who need to construct their own templates to produce anything but simple LaTeX code will undoubtedly need to consult Leslie Lamport's LaTeX manual [5] for more details of the system.

There are four sections. In the first section the overall structure of a LaTeX file is discussed whilst the second section is concerned with more detailed issues. In the third section we explain how to run the various programs that make up the LaTeX system. In the final section it is explained how to interpret any warnings or other error messages that may arise. (The use of Mathpad helps to reduce the amount of error when using LaTeX but offers no guarantee against error.) A more detailed description of error messages can be found in chapter 8 of Leslie Lamport's LaTeX manual.

It is useful to have the **latex2e** stencil on your screen whilst reading this appendix since all the templates referred to can be found there. Note that the latter stencil is rather large and what is described is but a very small selection thereof (the part that we feel will be relevant to all newcomers to LaTeX). Consult the help file for the **latex2e** stencil for more information.

A.1 Format of a LaTeX Document

A.1.1 Basic Format

It is common when preparing a large document for processing by LaTeX to split it into several files. One of these files is the so-called *root* file that contains general instructions about how to process the document as well as calls to other files that make up the document. We shall follow this practice in describing the format of a LaTeX document.

An example of a root file, prepared with Math∫pad, is shown in figure A.1. It is in fact a simplified version of the root file used to process this book, and in particular to process this appendix. It illustrates the typical format of a file to be processed by LaTeX.

The file begins with a so-called "preamble": all the lines from the first up to (but not including) "***Begin document***". The document proper is placed between "***Begin document***" and "***End document***". This general structure —first the preamble, then "***Begin document***", then the document proper and finally "***End document***"— *must* be adhered to. In addition there are rules governing what can be placed in the preamble and where, and similarly for the document proper. These rules are given later. For the moment we content ourselves with an overview of what is required.

The first line of our example root file tells the LaTeX system that the document is a book. (It also gives some additional information about how to print the document that we will ignore.) The effect of this command is to cause LaTeX to automatically add headings to each page and to number the pages in a certain way. The command also has other implications some of which we will discuss later.

The second line says that certain "packages" are needed to process the document. The "addfont" package is needed because the logo "Math∫pad" is not standard, and the "head" package is needed because the page headings in this book are also not standard. The details of the headings are specified by the call to the "MathSpadStyle" command

Book[12pt,*DraftOutput*]
Package{addfont,head}
\MathSpadStyle
Make index
Begin document
Title: Advice to Newcomers
Author: Hilary Backhouse
\MathSpadParameters
Front Matter
Table of Contents
Main Matter
Input File{Latexhelp}
Insert index here
End document

Figure A.1: A “Root” File Ready For Processing

that follows. We won’t go into the precise definition of this command at all; we include it only to give an example of where you should place instructions determining the overall appearance of your document should you too wish to diverge from the LaTeX standard.

The next command *Make index* instructs LaTeX to prepare an index. Note that this command doesn’t say where the index is to be placed. This is done by the command “*Insert index here*” in the document proper. (LaTeX does not create an index by default even when there are index entries in your document.)

As mentioned earlier, the document proper is bracketed by ***Begin document*** and ***End document***. The title and author form part of the document proper. In addition a date can be specified and an abstract can be added (but not the latter in the case of books).

Immediately following the title and author in our example root file there is a call to the “MathSpadParameters” command. This too specifies style conventions peculiar to this book, just like the “MathSpadStyle” command, the details of which are not relevant. What you do need to know is that LaTeX allows you to customize your document to your own taste; some of the parameters used to do so must be specified in the preamble and some in the document proper. (Again, the precise details are given later.)

A book generally contains “front matter”, “main matter” and “back matter”. The "front matter" usually includes things like the preface and the list of contents and is recognisable by the fact that the page numbers are in Roman rather than Arabic numerals. The two commands "*Front Matter*" and "*Main Matter*" in our example root file specify where these two sections begin. Thus the table of contents (whose position in the document is specified by the command "*Table of contents*" constitutes the front matter and the `Latexhelp` file constitutes the main matter. (In this example there is no back matter.)

The root file is terminated by the "***End document***" command. Anything after this in the file will simply be ignored by LaTeX.

```
\documentclass[12pt,draft]{book}
\usepackage{mathpad,makeidx,latexsym}
\input{mpmacros.tex}
\usepackage{addfont,head}
\MathSpadStyle
\makeindex
\begin{document}
\title{Advice to Newcomers}
\author{Hilary Backhouse}
\maketitle
\MathSpadParameters
\frontmatter
\tableofcontents
\mainmatter
\input{Latexhelp}
\printindex
\end{document}
```

Figure A.2: The "Root" File in Raw Form

If you are curious to know what you would have to enter if you were using raw LaTeX take a look at figure A.2. Note that there are several additional lines taken care of by the definition of the templates that have been used. More importantly, the use of the templates spares you the task of entering all those "\", "{" and "}" symbols, learning and remembering the precise syntax of each command, and ensuring that each is correctly spelt!

This concludes our overview of the format of a LaTeX document. In the following subsections we go into more details.

A.1.2 Preamble

Recall that every LaTeX document must begin with a *preamble*. The function of the preamble is to specify the basic style of the document.

The first statement of the preamble *must*[1] specify the class type of the document. This may be either *article*, *book*, *letter*, *report* or *slides*. The default settings of various style parameters affecting the finished output are determined by the class type chosen. The template DocumentClass is used to enter the class type. A brief description of the five document class types follows:

Article An article is usually a short document and has no chapters. By default, the title and abstract (if there is one) are not placed on separate pages.

Report This class is for longer documents and may have chapters. Both the title and abstract (if present) are placed by default on a separate page.

Book A book is considered to consist of *front matter*, *main matter* and *back matter*. The front matter may include such things as a copyright page, a foreword, a table of contents and so forth. The back matter may include a bibliography and an index. The main matter is the actual contents of the book. It is usual to number the pages of the front matter with Roman numerals and the main and back matter with Arabic numerals. It is also usual to omit the word "Chapter" in chapter headings in the front and back matter. To indicate to LaTeX where each of these three parts start, insert versions FrontMatter, MainMatter and BackMatter with template Book-DocClass at the appropriate places. If they are omitted everything is assumed to be main matter.

By default, the book document class will output left and right-hand pages Right-hand pages have an odd page number, left-hand pages have an even page number and chapters begin on a right-hand page. LaTeX assumes that this class of document does not have an abstract. Entering one into your document with the Abstract template will cause an error. The title is output by default on a separate page.

Letter This class is used for correspondence and should be used in combination with the Letter-DocClass template. There are versions with which to enter the sender's name and address, the opening greeting and so on.

Slides This is used when preparing slides. The fonts available are different and larger than with the other document classes and are designed to be read from a distance. Template Slides-DocClass is specially for use with this document class.

[1](See sending documents by email in the LaTeX manual for an exception.)

The class is specified by template DocumentClass and a version can be chosen with a place holder. This place holder may be filled by a list of one or more options separated by commas available via template ClassOptions. Some of the more commonly used options are:

10pt, 11pt, 12pt
The normal size of type in the finished document can be specified with one of these options. If none is selected then a ten-point size font will be used. These options are not available with the slides document class.

TwoColumn
A page will consist of two columns of text.

TwoSide Left and right-hand pages of output are made with the left pages numbered with odd numbers. This option can not be used with the slides document class and is the default with the book document class.

OneSide All pages have the same format. This is the default for all except the book document class.

leqno By default equation numbers are placed on the right but if this option is chosen then they will be placed on the left.

fleqn When this option is selected, displayed formulae are aligned at the left-hand side of the page instead of being centred. The distance at which they are aligned from the left margin is stored in the variable `mathindent`. A value can be assigned to this variable with template Parameters-OpsOn.

TitlePage, NoTitlePage
If the TitlePage option is selected then a title entered with the Title template and an abstract entered with the Abstract template will be set on separate pages. This is the default for all except the article class whose default is the NoTitlePage option. Note that the Abstract template should not be used with the book document class.

A4paper The output will be formatted to fit A4 size pages, i.e. 210mm $\times$ 297mm. Note that the conventional area for the text on the page seems to differ between America and Europe.

Following the class declaration, the preamble may contain a number of items. The order of these is not usually relevant but it is advisable to include any packages as soon as posssible after the DocumentClass command. For example, it may be necessary to load a *command file* (*macro file* in TeX jargon) or one of the many *packages* that are available. The need for command files and/or packages when using LaTeX in combination with Math∫pad arises if a non-standard LaTeX command is used in the definition of a template. The command file `mpmacros.tex` is an example: it is always needed if your Math∫pad document contains displayed expressions. It instructs LaTeX how to process such expressions in terms of the standard LaTeX commands. To load a command file use the InputFile template.

Packages are collections of commands designed to meet special-purpose needs. There is a large number available[2]. Template Packages enables you to include packages in your document.

If an index is to be generated then this must be indicated in the preamble using template Commands from the group **MakeIndex**. The more experienced LaTeX user may also wish to include in the preamble of a document definitions of LaTeX commands and/or environments. These can appear directly in the preamble or be read from another file. Use the Input File **T** version of template InputFile for the latter operation.

You may wish to split your document up into several files and then run LaTeX on just a selection of the files. The files to be processed should be listed using the IncludeOnly command which is to be found in the Preamble template. More details of the procedure for this can be found in section A.1.4.

Finally the preamble may include instructions about the style and layout of the document. For example whether running heads are required and where and in which style of numbering the pages should be numbered. Templates PageStyle and StylePageNo are used for this.

A list of the templates or versions which you are most likely to wish to use in the preamble is given below. Those which may *only* be used in the preamble are marked with a "†".

Packages † With this template a request to load one or more packages can be inserted into the preamble. Commands to load the *mathpad*, *latexsym*, *ifthen* and *makeidx* packages are not needed as these are automatically generated by the DocumentClass template.

PageStyle Each page consists of a header, the body of the page and the foot. With the PageStyle template the style of the header and foot can be altered. The first four versions of this template are called Plain, Empty,

[2] For a thorough account of the use of packages see [1]

Headings and MyHeadings. Plain, which is the default, has an empty head and the number centred at the foot of the page. With the Empty version both the head and foot of the page are left blank. If the Headings version is selected then the foot is empty and the head contains the page number and information determined by the document class. The user can specify the contents of the head and foot with the MyHeadings version. This option is for experienced LaTeX users and is not described here. See instead the LaTeX manual. The style of a particular page can be specified by inserting a version whose name begins with ThisPage within the text of the appropriate page. Certain LaTeX commands however, such as those to make chapter headings, contain page style instructions. Thus if a ThisPage ... version appears to have no effect, try repositioning it nearer the bottom of the page and after any sectioning commands.

StylePageNo The style of the page number can be set with this template. The possibilities available are Arabic numerals, upper or lowercase Roman numerals, and upper or lowercase letters.

Commands Version Make Index † from template Commands in the group **MakeIndex** must be included in the preamble if an index is to be generated.

InputFile Use the Input File **T** version to instruct LaTeX to process file **T**. (The difference between Input File **T** and Include File **T** is explained in section A.1.4.)

Preamble Template Preamble includes versions for several commands one of which has the name IncludeOnly †. This command allows the user to instruct LaTeX to process just a part of a document. Only those files called with an Include command whose names appear in the list entered with the IncludeOnly command will be processed. This is described in more detail in section A.1.4.

Another version which is sometimes useful is called No Auxiliary Files to be Written. Normally two or possibly three runs of LaTeX on your document are necessary to obtain the required output if there are cross-references to be filled in or such items as an index or table of contents are to be generated. During the first run particulars of the document are written to additional files and this information is used during the second pass to fill in cross-references, generate the index and so on. If version No Files Written is included in the preamble of the document then LaTeX will not write to any intermediary files but will only create new `dvi` and `log` files. The former file contains the output and the latter LaTeX's log of processing the document. Any existing intermediary files (`aux`, `toc`, `idx`, `glo`, `lof` and `lot` files) are left unchanged. The advantage of using No Auxiliary Files to be Written is that should a run of LaTeX be aborted then all the information accumulated during the first pass is not lost. (Of course it also means that the information in the files may become inaccurate.)

Parameters Several constraints on the layout of a page of output, such as the width and height of text on a page, are determined by the style parameters. The values assigned by default to them can be altered by using the Parameters-OpsOn template in conjunction with this template. See the on-line help file for the **latex2e** stencil for more details.

Page-Bottom This template has two versions, Ragged Bottom and Flush Bottom. Ragged Bottom allows the amount of text on a page to vary slightly from page to page. This is the default if LaTeX is formatting the output for printing on just one side of the page. Flush Bottom causes LaTeX to pad out the pages, if necessary, with extra vertical space so that the height of the text on each page is the same. This is the default if LaTeX is formatting the output for printing left and right handed pages.

A.1.3 Document Proper

The document proper must be separated from the preamble by the command ***Begin document*** which is entered with version Begin from the Begin/End(doc) template. It must also be terminated by the ***End document*** command which is to be found in the same template.

The user may make use of many of the templates provided in the **latex2e** stencil when preparing a document. On-line help information is provided for each template and so only a selection are discussed here.

Title This template can be used to enter the title, author(s) and date of the document. If the version of this template with which no date is set is used, then the date is automatically set to the current date.

Abstract An abstract can be entered with this template. However it should not be used to enter an abstract with a book document class as this will cause an error. An abstract in an article document class will appear where it is positioned in the text unless the TitlePage option was inserted into the list of class options. In this case it will appear on a page by itself.

InputFile The Input File **T** or Include File **T** versions are used to enter the contents of file **T** into a document at the position of the command. These commands are described in detail in section A.1.4. If an IncludeOnly command is present in the preamble, a file **T** that is called with the Include File **T** version will only be processed if its name is present in the argument of the IncludeOnly command. In addition an Include File **T** command may not appear in the preamble or in a file that is itself called by an Include command.

ContentsList The version TableOfContents should be entered at the position where a table of contents is required, version ListOfFigures where a list of figures is required and version ListOfTables where a list of tables is required. A second run of LaTeX is needed in order to produce these lists. See section A.3.

Ref(within doc) Some of the versions of the sectioning commands that can be entered via template SectionHeading include an optional parameter for a label. It is then possible to refer to the section without knowing how it will be numbered in the printed output. To refer to such a section, select the Ref(within doc) template and enter the label as the parameter to the Ref or (Ref) version. (The latter parenthesises the label in the output.) LaTeX will substitute the actual number of the section for the label in the output. Note that two runs of LaTeX on your document are necessary in order for LaTeX to be able to do this.

Other templates, such as Figure, Table and Equation/Display, also contain versions with a place holder for a label which can be used in the same way. Note that Ref and (Ref) only generate a number and if any text such as 'section' or 'equation' is needed then this must be entered by the user.

A particular place in the book can be referenced by inserting a label with the Label version at the required point and referring to it by entering the label into the parameter of the PageRef or (PageRef) versions.

SectionHeading Every sectioning command has a type (e.g. Chapter) and a heading. The type can be entered with template SectioningTypes. Associated with each type is a depth and according to the value of its depth the sectioning command is automatically numbered or not. If a number is not required choose a non-numbering version from the template. Also, according to the depth of the type and whether a table of contents has been requested an entry will be written to a file with extension `toc`. By default the heading is used as the table of contents entry. If this is not suitable, select the appropriate version from the template and enter a separate table of contents entry. Incidentally, if a table of contents entry is required but not generated, one can be made with version AddLineTOC of template ContentsList. Depending on the depth of the section type chosen and the page style set, the heading may also appear in the running head. Finally it is also possible to select versions in which a label can be entered. This label can be used to refer to the section. Note that this label does not appear in the output.

The values assigned by default that control whether a sectioning command is numbered or not and other style parameters can be changed with the Parameters-OpsOn template in combination with the Parameters template.

SectioningTypes Use this template to enter the type of sectioning command into the chosen version of SectionHeading.

A.1.4 Splitting the Document

When preparing a large document such as a book it is often convenient to split the text between several smaller files. The file containing the first instruction of the preamble is called the *root* file. The Input File **T** or Include File **T** commands from template InputFile can then be inserted into the root file at the place where the contents of file **T** are required. There are important differences between these two commands. An Input File **T** command may appear in the preamble and can also be placed within a file that is itself called by either an Input File **T** or Include File **T** command. An Include File **T** command, however, may neither be placed in the preamble nor within a file that is called by an Include File **T** command. More importantly, the text from a file **T** called with an Include File **T** command always begins on a new page and the text following that in file **T** will also start on a new page. If the text in each of such files should begin on a new page, such as if each file contains a complete chapter, then this isn't a problem but otherwise it could be.

The advantage of using an Include File **T** command is if you wish to run LaTeX on just a part of the document. (This can save you a lot of time just waiting for the processing to complete.) This might be the case if a change has only been made to one section for example. To do so insert into the preamble the IncludeOnly command (which is a version of the Preamble template). Into the placeholder of this command type the names of all of the files called with an Include File **T** command that you wish to process. If more than one name is to be entered, separate names from one another with a comma. If the document is now processed by LaTeX then all files called with an Include File **T** command that do not appear in the IncludeOnly command will not be processed. Note that the rest of the document is processed as if these files had not been omitted. Thus if a complete chapter is omitted, for example, the numbers of succeeding chapters do not alter. LaTeX does not read the omitted files but assumes that no changes have been made to them since the last time they were processed.

A.2 Typing the Document

Certain keyboard characters and combination of characters have a special meaning to LaTeX. Others are handled in a way that may be different from that which you are accustomed to. Thus, if Math∫pad is being used to generate output for processing by LaTeX, then the document must be typed according to LaTeX's conventions.

A.2.1 White Space

Horizontal Space The characters obtained by pressing the space bar or the **return** key are considered by LaTeX to be the same space character. Any sequence of space characters is considered to be a sequence of just one space character. Thus if you wish to increase the space between two typed items it does not help to insert extra spaces with the space bar between them. The extra spaces will be shown on the screen but will be ignored by LaTeX and so will not appear in the final output. To insert extra space use instead the template Horizontal (from group **Spacing**) or the raw LaTeX command \␣ where ␣ stands for the character obtained by pressing the space bar. The special symbol ~ is also used to insert space but in places where it would be inappropriate to break the line. For example, I would input my name as H.$\sim$Backhouse so that the "H." doesn't appear at the end of a line by itself.

When text is set in a typewriter-like font the inter word space is a fixed length causing a ragged right-hand or, less usually, left-hand margin. Except when such a font is specified or it has been instructed to set the typing with a ragged margin, LaTeX will stretch or shrink the inter word space on each line so that the right margin is straight. Thus inserting space by the methods mentioned in the previous paragraph may not have the desired effect. To insert an absolute length of space use instead versions space{length} or space{length}* of template Horizontal and fill in the length required (not forgetting the units). Version space{length} will not produce any space at the start or end of a line (of output) but version space{length}* will.

Vertical Space A completely blank line in the text indicates to LaTeX the end of a paragraph. Multiple blank lines entered by means of the **return** key are considered by LaTeX to be just one blank line. One or more blank lines immediately after a sectioning command such as chapter or section will also be ignored. Note that a blank line appearing on the screen which has been generated by using a template may not necessarily generate a blank line in the corresponding LaTeX code.

Extra vertical space can be added after a line break by using the \\[length] version from template LineBreak. Insert into the place holder the length of the extra vertical space required. (Don't forget the units of length.)

It's also possible to add extra vertical space with template Vertical (from group **Spacing**). If a version is inserted between paragraphs then the extra vertical space will appear there. If it is inserted in the middle of a line (of output) the extra space is added immediately after the end of that line of output.

A.2.2 Dashes

Typing one, two or three hyphens after each other will produce different lengths of dashes as follows: "`-`" produces a hyphen as in X-ray,"`--`" produces an "en-dash" which is commonly used to separate numbers as in 10–20, and "`---`" produces an "em-dash" —which is used for a parenthetical or explanatory remark such as this one—.

A.2.3 Full Stops

It is usual to set some extra space after the full stop at the end of a sentence. To indicate to LaTeX that a full stop is at the end of a sentence follow it either by a space or a **return** character. LaTeX assumes that all full stops followed by a space mark the end of a sentence unless they are preceded by an uppercase letter. To insert a period followed by a space in other situations such as the second period in the abbreviation "i.e." use the template Punctuation+Space. Some of the more commonly used abbreviations, for example e.g., are to be found in the template Abbreviations and can be entered into the document directly from there. Note that this template adds a space after the abbreviation and so could not be used in the preceding sentence where "e.g." is followed by a comma.

Should a sentence end with an uppercase letter enter the period with the appropriate version of the template Punctuation+Space.

A.2.4 Special Characters

The ten characters

% & ~ # $ \ { } _ ^

acquire special meanings in LaTeX and so cannot be entered directly from the keyboard. These must be entered instead with template SpecialChars. They will probably all cause problems if they are inadvertently inserted via the keyboard but some warrant a special mention.

% This is used for adding comments. LaTeX will ignore the % and all the following characters up to and including the character marking the end of the line (in the `tex` file).

\ All LaTeX commands begin with this character which is called a *backslash*. Thus, if the backslash is followed by a sequence of characters that is not a defined LaTeX command an error will be reported, but if followed by characters that do form a command then that command will be executed!

$ A dollar sign changes the mode either out of or into math mode depending on whether LaTeX is or isn't currently in math mode. An unexpected change of font to italic type may be caused by the occurrence of a spurious dollar sign in the text.

Certain non-alphanumeric symbols following a backslash are also LaTeX commands. A few of these commands are listed in table A.1 as an aid to debugging LaTeX errors.

Symbols	Meaning
\(	Enter math mode
\)	Leave math mode
\[	Enter displayed math mode
\]	Leave displayed math mode
\\	Start a new line
l	Start a new line and insert vertical space of length *l*
\␣	Enter a space
\> \< \=	LaTeX tabbing commands
\+ \- \` \'	LaTeX tabbing commands

Table A.1: Some LaTeX commands.

A.2.5 Linebreaks

One of the most important (and difficult) tasks undertaken by LaTeX (to be more precise TeX) is breaking the text into lines and into pages. The difficulty of the task is evident from the fact that it often doesn't succeed!

Bad line breaks may be caused by the presence of a word which LaTeX won't or doesn't know how to hyphenate. In these cases use the Hyphenate template to give advice on hyphenation. LaTeX will only hyphenate an already hyphenated word at the existing hyphen which may result in line breaking problems if such a word occurs at the end of a line. The only workable solution is to rewrite the offending sentence (or one in the immediate vicinity).

In other cases the user can insert the command LineBreak from template LineBreak into the text forcing LaTeX to break the text at the points at which it is placed. Another version of the command has an argument with can take the value 0, 1, 2, 3 or 4. The value given indicates how strongly you wish the text to be broken at the point in question. LineBreak[4] forces a break whereas LineBreak[0] indicates no particular wish for a break. The only difference that the presence of LineBreak[0] makes is that a break can be made at the point even if it is not allowed in normal circumstances, for example in an already hyphenated word.

The commands NoLineBreak and NoLineBreak[T] are similar. Both NoLineBreak and

NoLineBreak[4] prohibit a line break at the places at which they are placed, and a NoLineBreak[0] is the same as LineBreak[0].

Line breaks can also be made with the commands \\ and NewLine from template LineBreak. These two commands, unlike the LineBreak and LineBreak[**T**], do not force the line to be aligned at the right with the right-hand margin.

Occasionally LaTeX puts a line break somewhere where you don't want one, for example, in the middle of a name. This can be prevented by entering the piece of text which mustn't be split, into the place holder of template UnbreakableText.

A.2.6 Pagebreaks

Similarly, LaTeX fails sometimes to find a good place at which to split the text into pages. This usually results in an insufficiently full page and an "`Underfull \vbox`" warning message being output. In section A.4.2 there are some suggestions on how this problem may be solved. If they fail you can insert the command NoPageBreak to (try to) prohibit a page break or PageBreak to cause a page break. Both these commands are to be found in the template PageBreak. Like the commands NoLineBreak and LineBreak they may take an optional argument to indicate the strength of the desire to break or not to break a page at this point. If the PageBreak and NoPageBreak are positioned between two paragraphs then they apply to that point. Otherwise they apply to the end of the line in the LaTeX output in which they are placed.

Alternatively you can increase the height of text allowed on a page in order to put more text than is normal on the page. To do this, insert the command Enlarge[**T**] from template PageBreak above the point where LaTeX starts a new page. Set **T** to a suitable length and then insert command PageBreak at the point where you want a page break. Remember that texts on facing pages with two-sided printing are usually the same height.

You can also cause a page break by inserting the command NewPage from template PageBreak. Unlike the PageBreak command the NewPage command will not cause the text on the page to be spaced out to reach the bottom of the page.

Fixing bad page breaks can be (very) time consuming so it's best to leave them until you are definitely preparing the final version of your document.

A.2.7 Creating an Index

There is a program *MakeIndex* available which can be used in conjunction with LaTeX to produce an index. As this is a relatively simple way of producing an index we only discuss this method here. If you don't wish to make use of the *MakeIndex* program consult the LaTeX manual on how to generate an index yourself.

In order to use the *MakeIndex* program, packages *makeidx* and *ifthen* must be loaded into the preamble of the document. If you have entered template DocumentClass from the **latex2e** stencil into your document then these packages are automatically included in your document. (If you didn't use template DocumentClass then insert template Packages into the preamble and enter *makeidx, ifthen* into its place holder.)

The templates needed for the compilation of an index can be found in the group labelled **MakeIndex**. To generate an index you must not only indicate the items in the document which are to be indexed but also inform LaTeX that you wish to make an index and specify where in the document it is to appear. To do so, first enter version Make Index from template Commands into the preamble of the document. This instructs LaTeX to write details of the index entries in the document to a file with extension `idx`. Then enter version Insert Index from the same template at the place in the document at which you wish the index to appear. This command instructs LaTeX to process at this point the file with extension `ind` of sorted index entries (if it exists).

We are now ready to discuss the generation of the index items themselves. This is done with templates Entry and SEE-Entry both of which have two versions.

An index item may be a normal index entry with a page number or it may be an index entry which refers the reader to another index entry. The latter we call a "SEE" entry. As the number of the page in which a "SEE" entry is inserted into the document is irrelevant, it is a good idea to put all "SEE" entries together (for example in a separate file). These are entered with version enter SEE entry of template SEE-Entry. Once such an item has been entered use the Version button on the console to change the version to display SEE command. Only a selection of the index fields is then displayed in the document window improving the readability. Similarly enter a standard index item with the enter index item version from template Entry and then hide it to improve the appearance of your document on the screen by changing to version hide index command.

A detailed description of how to enter details into the enter SEE entry and enter index item versions is given in the help file associated with the templates.

After the document has been processed by LaTeX, *MakeIndex* is used to process the information in the `idx` file and write the sorted index entries to an ind file. On subsequent runs of LaTeX the ind file is used to generate an index. More detailed instructions on this are given in section A.3.1.

A.2.8 Bibliography

Just as there is a program available to assist with the making of an index, so is there a program, BibTeX, which can be used in conjunction with LaTeX to produce a bibliography. We only discuss this method of producing a bibliography here. The templates that you will need can be found in the group labelled **BIBTEX**.

To use this program you must first enter all your book references into a database. Although this is time consuming you can use the same database for other documents and share it with colleagues.

Use the DocTypes and OptFields templates to enter your references into an `mpd` file. This entails inserting for each entry details such as the type of publication, the name of the author and so on together with a search key. As this is quite complicated see the help file for advice on how to do it. When all the entries have been entered into the database in the document window (`mpd` file) you must generate the version of the database file which is to be processed by BibTeX. This must have a `bib` extension. Thus, after clicking on the Output button at the top of the document window *edit the selection in the file selector window* to ensure that the extension of the file to which the LaTeX code is written is `bib`.

The search key given to each reference in the database must be unique. Where you wish to cite a publication in the text of your document select Cite from template Ref(other doc) and insert the search key of the required reference into the place holder. If you wish to add a comment such as a page number to the citation use version Cite[Ref,Comment] instead.

At the point in the document where you wish the list of references to appear insert command Bibliography filling its place holder with the name(s) of your database(s). These are assumed to have a `bib` extension. Also insert into your document file a version from the template BibliographyStyle to indicate the style required for the list of references. This may be one of the following:

Plain Entries are sorted alphabetically and labelled with a number.

Unsrt Entries are in the order they appear in the text and are labelled with a number.

Alpha Entries are sorted alphabetically but the label is formed from the author's name and the year, e.g. [BAC95].

Abbrv The same as plain except that the entries contain more abbreviations.

The bibliography itself can now be generated. See section A.3 on how to do this.

A.3 Running LaTeX

In order to run LaTeX to produce your final output you must produce a `tex` file for each of your document (`mpd`) files. As described more fully in section 5.1, a document's `tex` file is produced by selecting `Output` at the top of the document window into which it has been loaded, specifying via the file selector (if necessary) the directory to which it is to be written and then clicking on `OK`. Now you are ready to run LaTeX.

Move the cursor to a system window and ensure that the current directory is the directory containing your `tex` files. Suppose that `Root.mpd` is the Math∫pad document file that contains the preamble. Then type the command

```
latex Root
```

and LaTeX should begin to process the file `Root.tex`. Various messages will appear on the screen indicating how LaTeX is progressing. An example of such messages is shown in table A.2. In it the root document file is called `Guide` which contains a call to another document file, `Latexhelp`. LaTeX begins processing `Guide.tex` on line 2, it is reading the style file for a book on line 4 and in lines 5–17 it processes the calls to the other packages and style files that were specified in the preamble. On line 18, LaTeX opens file `Guide.idx` to which index entries are to be written. On line 19, it is reading information from `Guide.aux` and `Latexhelp.aux` which were output during a previous run of LaTeX. The `aux` files contain information that is needed for, among other things, filling in cross-references. On line 21, it begins processing file `Latexhelp.tex` where it encounters its first chapter command. The numbers between square braces are the numbers on pages that have been output to `Guide.dvi` and it can be seen from these that the warning that there is insufficient text on a page applies to page 3. On line 25, the index items are read from file `Guide.ind` and new versions of the `aux` files written. Diagnostics are written to `Guide.log`.

Note that it is not necessary to instruct LaTeX to process anything other than the root file.

Messages sent to the screen	Line
This is TeX, Version 3.1415 (C version 6.1)	1
(Guide.tex	2
LaTeX2e <1994/12/01> patch level 1	3
(/usr/TeX/lib/texmf/tex/latex2e/book.cls	4
Standard Document Class 'book' <1994/12/09>.	5
(/usr/TeX/lib/texmf/tex/latex2e/bk12.clo))	6
(/home/roland/tex/inputs/mathpad.sty)	7
(/usr/TeX/lib/texmf/tex/latex2e/makeidx.sty)	8
(/usr/TeX/lib/texmf/tex/latex2e/ifthen.sty	9
Package: 'ifthen' v0.1b <1993/12/17> (DPC)	10
English documentation <1993/12/17> (DPC)	11
) (/usr/TeX/lib/texmf/tex/latex2e/latexsym.sty	12
Package: 'latexsym' v2.1a <93/12/06> (FMi)	13
English Documentation <93/12/06> (FMi))	14
(/home/roland/tex/inputs/mpmacros.tex)	15
(/home/roland/tex/inputs/addfont.sty)	16
(head.sty)	17
Writing index file Guide.idx	18
(Guide.aux (Latexhelp.aux))	19
(/usr/TeX/lib/texmf/tex/latex2e/Ulasy.fd)	20
(Latexhelp.tex	21
Chapter 1.	22
(Umathss.fd) [1] [2]	23
Underfull vbox (badness 5607) has occurred while output is active	24
[3] [4] [5] [6]) [7] (Guide.ind) (Guide.aux (Latexhelp.aux)))	25
(see the transcript file for additional information)	26
Output written on Guide.dvi (7 pages, 20784 bytes).	27
Transcript written on Guide.log.	28

Table A.2: A run of LaTeX.

As can be seen from table A.2, LaTeX reads from and writes to several other files when processing a document. Providing a No Auxiliary Files to be Written command has not been inserted into the preamble of your document, then for each `mpd` file an `aux` file is read (if it exists) and written to. Any other intermediary and diagnostic files that are used or created have the same name as the root file but with a different extension. A list of possible file extensions is given in table A.3. Note that some of these files are only generated if the necessary commands are present in the document's `tex` files and some are generated by programs *MakeIndex* and BibTeX.

If information is written to a file which is processed on a subsequent run of LaTeX, such as a table of contents, then to obtain the complete output of your document it is necessary to run LaTeX twice and possibly three times in succession on your root file. If

Ext.	**Contents of file**
`aux`	Additional information necessary for such things as cross-referencing.
`dvi`	Output from LaTeX program.
`log`	Diagnostics from LaTeX program.
`toc`	Items for table of contents. Generated by a version of template ContentsList.
`lof`	List of figures. Generated by a version of template ContentsList.
`lot`	List of tables. Generated by a version of template ContentsList.
`idx`	List of index items output if version Make Index from template Commands (group **MakeIndex**) is in the preamble. The `idx` file must be processed by program *MakeIndex*.
`ind`	List of index items processed by *MakeIndex* used by LaTeX to make an index at the point where version Insert Index from template Commands (group **MakeIndex**) appears in document.
`ilg`	Diagnostics from *MakeIndex* program.
`glo`	List of glossary items.
`bib`	Database of references to be used by BibTeX.
`bbl`	List of bibliography items output by BibTeX and read by LaTeX.
`blg`	Diagnostics from BibTeX program.

Table A.3: Extensions of files created and/or used by LaTeX, BibTeX and *MakeIndex*.

a LaTeX run is aborted for any reason then the contents of the files which it outputs in order to read in again on a subsequent run will have probably been lost. Information that LaTeX requires will be not available on the following run. This can result in, for example, an empty table of contents and many warning messages about missing references. This is rectified by running LaTeX again on the root file. Alternatively command No Auxiliary Files to be Written can be entered into the preamble with the Preamble template. LaTeX will then only write new versions of the `dvi` and `log` files.

A.3.1 Index

If an index is to be produced with the aid of *MakeIndex* begin as before by running LaTeX on `Root`. A file `Root.idx` will be produced. Next run *MakeIndex* by typing

```
makeindex Root
```

This will generate a file, `Root.ind`, containing the sorted index entries. Any error or warning messages will be sent to the screen and also to file `Root.ilg`. Finally rerun LaTeX on `Root`. LaTeX will read in and process file `Root.ind` and an index should now appear in your document. Unless there is a No Auxiliary Files to be Written command in your preamble a new `Root.idx` will also have been generated. However, unless a change has been made to the document since the previous run, this file will have the same contents as before. Note that if ever a change is made to the document it is necessary to repeat the whole process to be sure that the index is correct. That is, run LaTeX, then `makeindex` and finally LaTeX again on `Root`.

LaTeX will output the warning message "`No file Root.ind.`" if file `Root.ind` does not exist.

A.3.2 Bibliography

Generation of a bibliography with BibTeX is similar. First run LaTeX on your root file. This time however, the information to be processed is written to `Root.aux` and not to a special file. Then execute BibTeX with the command

```
bibtex Root
```

and file `Root.bbl` will be produced. It is now necessary to run LaTeX *twice* more on `Root` to obtain a complete bibliography.

A.3.3 Output

In order to display your output in `Root.dvi` on the screen or to send it to a printer consult your local expert. The preview command which works on my machine is `xdvi Root` .

A.4 Messages

In the course of execution, all three programs (LaTeX, BibTeX and *MakeIndex*) generate messages. Some simply indicate what is happening, some are warnings and some are error messages. Error messages suspend execution and require your intervention before processing can continue.

Wouldn't it be wonderful if we could say that using Math∫pad the error messages are a thing of the past! Unfortunately that is not the case. Some types of error messages that plague the conventional LaTeX user (such as unbalanced curly braces, spurious dollar signs and misspelt commands) are reduced very substantially in number but you will inevitably be confronted with both warnings and error messages at some stage. This section explains how to deal with the ones that are most likely to arise. Messages not discussed here will probably have been caused by a badly defined template, your own use of raw LaTeX or some complex combination of circumstances that causes LaTeX to malfunction.

A.4.1 Error Messages

```
! I can't find file '...'
Please type another input file name:
```

If the extension of the file specified is `cls` or `sty` then LaTeX can't find either a style or package file. If this is not caused by a typing error ask advice from a colleague. If however the missing file has extension `tex` then the error was probably caused by one of the following:

- The file name was incorrectly typed. Retype it and press the **return** key.
- The required file is not in the current directory. Abort LaTeX by typing **C-D**. Move to the correct directory and try again. If you are in the correct directory

but the file is not there then return to Math∫pad and repeat the output instruction for the required file. (That is click on Output at the top of the document window.) The directory to which the output file will be sent is specified at the top of the file selector. If this is not the correct one either edit it to the correct one and select OK, or select Cancel, open the default window and specify the correct output directory and then repeat the output instruction. You can then return to the prompt window and try running LaTeX again.

- The `tex` file is not there because you forgot to instruct Math∫pad to make an output file. Return to Math∫pad and make one.

  Remember that if a file contains calls to other files an output file for each must be generated.

`! Missing \begin{document}..`

There is no ***Begin document*** command in your document. This should be placed at the end of the preamble, immediately before the document proper.

Alternatively there is an instruction in the preamble which may only be positioned in the document proper. Move the offending instruction to after the ***Begin document*** statement.

`! Can be used only in preamble.`

There is an instruction after the ***Begin document*** statement which may only be used in the preamble. Move the offending instruction to before the ***Begin document*** statement.

`! There's no line here to end.`

LaTeX has found a newline command (either NewLine or \\ from template LineBreak or the LaTeX commands \\ or \newline) at an incorrect position. Remove it. If you were trying to add extra vertical space see section A.2.1 on white space.

`!Illegal unit of measure (pt inserted).`
`!Missing number treated as zero.`

LaTeX was expecting a length. You have probably forgotten to specify the units.

`! Undefined control sequence`

This has almost certainly been caused by entering \ into your document via the keyboard instead of via the template SpecialChars.

After LaTeX has detected an error it prompts the user with a “?” and waits for further instructions. It is often a good idea to instruct LaTeX to proceed in order to find as many errors as possible in one run. To proceed press the **return** key. If you wish to stop LaTeX type **i \ stop** or **x**. The latter will stop LaTeX immediately whereas the former will output the last page of output up to the error.

Sometimes LaTeX will output a “*” and stop without an error message.

This is probably caused by a missing ***End document*** in your Mathpad document. Abort LaTeX by typing `\stop`, return to Mathpad and enter it from template Begin/End(doc).

A.4.2 Warning Messages

LaTeX may output many warning messages. These do not cause LaTeX to stop but indicate that there is something wrong with the output. Not only do the messages appear on the screen but they are also written to the file, `filename.log`, where they can be studied at leisure. Most warnings are caused by missing cross-references or problems with the layout.

The first type of warning is usually easily fixed. It may just be necessary to rerun LaTeX so that it has an up to date `aux` file that it can use. (See section A.3 on running LaTeX.) Alternatively a label may have been omitted or a label or the argument of a reference has been misspelt.

A warning message will also appear if LaTeX has had to put too much or too little on a line or a page. This can often be problematic. It may even result in having to rewrite bits of the text. If this is the case, do not do so until preparing the final version of the document to avoid altering the text unnecessarily.

`Overfull \hbox ...`

LaTeX has put too much text on the line. (If the option DraftOutput has been entered into the class type command then the end of this line is marked with a black box.) This is often caused by problems with hyphenation. It could be that LaTeX does not know how to hyphenate something in the document. Use template Hyphenate to give instructions on hyphenation. Also, LaTeX will only hyphenate a hyphenated word at the existing hyphen. Thus this warning can be caused by the presence of such a word at the end of a line. In this case it's probably easiest just to rewrite the sentence.

See also section A.2.5 on linebreaks.

`Underfull \hbox ...`

This has probably been caused by trying to insert vertical space into the page by outputting an empty line. It isn't a problem but is a nuisance if you have too many such warnings appearing on the screen — you can miss seeing important warning messages.

The problem is usually caused by an extra `\newline` or \\ in your `tex` file. If you've entered two consecutive newline commands in your document, that is NewLine or \\ from the template LineBreak or the LaTeX commands \\ or `\newline`, replace them by the template version \\[length], a version from the Vertical (from group **Spacing**) template or by different LaTeX commands. (See section A.2.1 on vertical white space.) If the problem is caused by a template you will probably have to ask its author to change it for you.

See also section A.2.5 on line breaks.

`Underfull \vbox ...`

LaTeX has had difficulty splitting the document into pages and has output a page with insufficient text on it. LaTeX will try to put the extra vertical space around displayed items and above and below sectioning commands so that in reality the page may be acceptable. Often however it is not.

It may be that you have a large item in your document which LaTeX is forbidden to split between pages such as text entered with the Tabular template. One solution is to make the offending item into a so-called "float" with either the Table or Figure templates. Alternatively, use a template which can be split between pages to enter the item. For example, use the Tabbing templates in place of the Tabular ones.

For more suggestions see section A.2.6 on page breaking.

`Overfull \vbox ...`

Again LaTeX has had difficulty splitting the document into pages but this time has put too much on a page. The problem may be solved as for the previous warning by entering a large item, which may not be split between pages, into the document in a different manner.

For more suggestions see the section on page breaking.

Appendix B

LaTeX commands and environments with associated templates.

by Hilary Backhouse

This appendix lists LaTeX2$_\varepsilon$ commands that are included in the **latex2e** stencil. The first column lists LaTeX2$_\varepsilon$ commands, the second column a template in which it occurs, and the last column the section of the stencil in which the template can be found. (Note that some commands appear in more than one template.)

The division of the **latex2e** stencil into sections was necessitated by the stencil's large size: it can be difficult to find the particular template that you require. The content of each section has been designed so that, wherever possible, the templates in each group have a common theme. A dummy template at the start of each group generates a bold heading for the group within the stencil. The groups are as follows:

- Preliminaries—items normally appearing in the preamble.
- Document—particular document classes and the Begin/End(Doc) template.
- Miscellaneous—templates corresponding to LaTeX commands which can not be placed in any of the other groups.
- Boxes—"boxes" are items which are treated as a single indivisible unit by LaTeX.
- Counters
- Definitions—user-defined commands.

- Floats—generate items which can "float" from one position to another in the text.
- Fonts
- Lists
- Mathematics
- Misc templates—templates for general use not associated with any LaTeX commands.
- Pictures
- Spacing
- Tabular and Tabbing
- BIBTEX—templates for creating a bibliography.
- MakeIndex—templates for creating an index
- Graphics package
- ifthen package

Help with the use of each template can be obtained by clicking on the template with the far mouse button and then selecting Help from the menu. The help for each group of templates is stored together in one document (`mpd`) file. If you click on the Help menu item in the dummy heading template then the help file is opened at the start of the help for that group of templates. Sometimes you will find information at this point that is relevant to all the templates in the group.

LaTeX COMMAND	TEMPLATE	GROUP
`\.`	Accent	Miscellaneous
`^ superscript`	Scripts	Mathematics
`^ superscript`	ScriptText	Miscellaneous
`_ subscript`	Scripts	Mathematics
`_ subscript`	ScriptText	Miscellaneous
`\␣`	Horizontal	Spacing
`\␣`	Punctuation+Space	Miscellaneous
`\^`	Accent	Miscellaneous
`\^`	SpecialChars	Miscellaneous
`\"`	Accent	Miscellaneous
`\~`	Accent	Miscellaneous
`\#`	SpecialChars	Miscellaneous
`\$`	SpecialChars	Miscellaneous
`\%`	SpecialChars	Miscellaneous
`\&`	SpecialChars	Miscellaneous
`~`	SpecialChars	Miscellaneous
`_`	SpecialChars	Miscellaneous
`\{`	SpecialChars	Miscellaneous
`\}`	SpecialChars	Miscellaneous
`\\`	LineBreak	Miscellaneous
`\\`	Poetry	Miscellaneous
`\\*`	LineBreak	Miscellaneous
`\\*`	Poetry	Miscellaneous
`\-`	Hyphenate	Miscellaneous
`\:`	Space	Mathematics
`\;`	Space	Mathematics
`\!`	Space	Mathematics
`\,`	Space	Mathematics
`\,`	Horizontal	Spacing
`\'`	see `\a'`	
`` \` ``	see `` \a` ``	
`\=`	see `\a=`	
`\=`	Tabbing-Row(DIY)	Tabular and Tabbing
`\=`	Tabbing-KillLine	Tabular and Tabbing
`\+`	Tabbing-Row(DIY)	Tabular and Tabbing
`\-`	Tabbing-Row(DIY)	Tabular and Tabbing
`\<`	Tabbing-Row(DIY)	Tabular and Tabbing
`\>`	Tabbing-Row(DIY)	Tabular and Tabbing

\>	Tabbing-Row	**Tabular and Tabbing**
\@.	Punctuation+Space	**Miscellaneous**
\@?	Punctuation+Space	**Miscellaneous**
\@!	Punctuation+Space	**Miscellaneous**
\@:	Punctuation+Space	**Miscellaneous**
10pt	ClassOptions	**Preliminaries**
11pt	ClassOptions	**Preliminaries**
12pt	ClassOptions	**Preliminaries**
\a'	Accent	**Miscellaneous**
\a`	Accent	**Miscellaneous**
\a=	Accent	**Miscellaneous**
a4paper	ClassOptions	**Preliminaries**
a5paper	ClassOptions	**Preliminaries**
abbrv	BibStyle	**BIBTEX**
\abovedisplayshortskip	Parameters	**Preliminaries**
\abovedisplayskip	Parameters	**Preliminaries**
abstract environment	Abstract	**Miscellaneous**
\acute	Accents	**Mathematics**
\addcontentsline	ContentsList	**Miscellaneous**
address	OptFields	**BIBTEX**
\address	Letter-DocClass	**Document**
\addtime	Slides-DocClass	**Document**
\addtocontents	ContentsList	**Miscellaneous**
\addtocounter	OpsOn	**Counters**
\addtocounter	Parameters-OpsOn	**Preliminaries**
\addtolength	LengthOps	**Miscellaneous**
\addtolength	Parameters-OpsOn	**Preliminaries**
\addvspace	Vertical	**Spacing**
alltt	PackageNames	**Preliminaries**
\alph	PrintCtr(SpecifyStyle)	**Counters**
\alph	SetDefaultPrintStyle	**Counters**
\Alph	PrintCtr(SpecifyStyle)	**Counters**
\Alph	SetDefaultPrintStyle	**Counters**
alpha	BibStyle	**BIBTEX**
amstex	PackageNames	**Preliminaries**
\and	Title	**Miscellaneous**
\and	operator $\wedge$	**ifthen package**
annote	OptFields	**BIBTEX**

\appendix	Appendix	**Miscellaneous**
\arabic	PrintCtr(SpecifyStyle)	**Counters**
\arabic	SetDefaultPrintStyle	**Counters**
array environment	Array-Enter	**Mathematics**
article	DocTypes	**BIBTEX**
article	DocumentClass	**Preliminaries**
author	OptFields	**BIBTEX**
\author	Title	**Miscellaneous**
\b	Accent	**Miscellaneous**
b5paper	ClassOptions	**Preliminaries**
babel	PackageNames	**Preliminaries**
\backmatter	Book-DocClass	**Document**
\backslash	SpecialChars	**Miscellaneous**
\bar	Accents	**Mathematics**
\baselinestretch	Parameters	**Preliminaries**
\begin{document}	Begin/End(doc)	**Document**
\belowdisplayshortskip	Parameters	**Preliminaries**
\belowdisplayskip	Parameters	**Preliminaries**
\bibindent	Parameters	**Preliminaries**
\bibitem	Bibliography(DIY)	**Miscellaneous**
\bibliography	Bibliography	**BIBTEX**
\bibliographystyle	BibStyle	**BIBTEX**
BIBTEX	LaTeX	**Miscellaneous**
\bigcap	VariableSizedSymbols	**Mathematics**
\bigcup	VariableSizedSymbols	**Mathematics**
\bigodot	VariableSizedSymbols	**Mathematics**
\bigoplus	VariableSizedSymbols	**Mathematics**
\bigotimes	VariableSizedSymbols	**Mathematics**
\bigskip	Vertical	**Spacing**
\bigskipamount	Parameters	**Preliminaries**
\bigsqcup	VariableSizedSymbols	**Mathematics**
\biguplus	VariableSizedSymbols	**Mathematics**
\bigvee	VariableSizedSymbols	**Mathematics**
\bigwedge	VariableSizedSymbols	**Mathematics**
\boldmath	MathStyle(bold)	**Fonts**
book	DocTypes	**BIBTEX**
book	DocumentClass	**Preliminaries**
booklet	DocTypes	**BIBTEX**

booktitle	OptFields	**BIBTEX**
\boolean	Boolean	**ifthen package**
\bottomfraction	FloatParameters	**Floats**
bottomnumber	FloatParameters	**Floats**
\breve	Accents	**Mathematics**
\c	Accent	**Miscellaneous**
\caption	Figure	**Floats**
\caption	Table	**Floats**
\cc	Letter-DocClass	**Document**
\cdot	Dots	**Mathematics**
\cdots	Dots	**Mathematics**
center environment	Layout-c,l,r	**Miscellaneous**
chapter	OptFields	**BIBTEX**
chapter	LaTeXCtrs	**Counters**
chapter	ContentsList-SecTypes	**Miscellaneous**
\chapter	SectioningTypes	**Miscellaneous**
\chapter*	see SectionHeading	**Miscellaneous**
\check	Accents	**Mathematics**
\circle	Objects	**Pictures**
\circle*	Objects	**Pictures**
\cite	Ref(other doc)	**Miscellaneous**
\cleardoublepage	PageBreak	**Miscellaneous**
\clearpage	PageBreak	**Miscellaneous**
\cline	Tabular-Line	**Tabular and Tabbing**
clock	ClassOptions	**Preliminaries**
\closing	Letter-DocClass	**Document**
color	PackageNames	**Preliminaries**
\color	Color	**Miscellaneous**
\colorbox	Color	**Miscellaneous**
\columnsep	Parameters	**Preliminaries**
\columnseprule	Parameters	**Preliminaries**
\coprod	VariableSizedSymbols	**Mathematics**
crossref	OptFields	**BIBTEX**
\d	Accent	**Miscellaneous**
\dashbox	Boxes	**Pictures**
\date	Title	**Miscellaneous**
\dblfloatpagefraction	FloatParameters	**Floats**
\dblfloatsep	FloatParameters	**Floats**

\fbox	BoxMandF	**Boxes**
\fboxrule	BoxParameters	**Boxes**
\fboxsep	BoxParameters	**Boxes**
\fcolorbox	Color	**Miscellaneous**
figure	LaTeXCtrs	**Counters**
figure environment	Figure	**Floats**
figure* environment	Figure	**Floats**
final	ClassOptions	**Preliminaries**
fleqn	ClassOptions	**Preliminaries**
\floatpagefraction	FloatParameters	**Floats**
\floatsep	FloatParameters	**Floats**
\flushbottom	Page-Bottom	**Miscellaneous**
flushleft environment	Layout-c,l,r	**Miscellaneous**
flushright environment	Layout-c,l,r	**Miscellaneous**
\fnsymbol	PrintCtr(SpecifyStyle)	**Counters**
\fnsymbol	SetDefaultPrintStyle	**Counters**
footnote	LaTeXCtrs	**Counters**
\footnote	Footnote	**Miscellaneous**
\footnotesep	Parameters	**Preliminaries**
\footnotesize	TextSize	**Fonts**
\footskip	Parameters	**Preliminaries**
\frac	Fractions	**Mathematics**
\frame	Boxes	**Pictures**
\framebox	BoxMandF	**Boxes**
\framebox	Boxes	**Pictures**
\frenchspacing	FrenchSpacing	**Miscellaneous**
\frontmatter	Book-DocClass	**Document**
graphics	PackageNames	**Preliminaries**
graphpap	PackageNames	**Preliminaries**
\graphpaper	Commands	**Pictures**
\grave	Accents	**Mathematics**
\H	Accent	**Miscellaneous**
\hat	Accents	**Mathematics**
\headheight	Parameters	**Preliminaries**
headings	PageStyle	**Preliminaries**
\headsep	Parameters	**Preliminaries**
\height	BoxDimensions	**Boxes**
\hfill	FillGap	**Miscellaneous**

`\kill`	Tabbing-Row(DIY)	**Tabular and Tabbing**
`\kill`	Tabbing-Enter	**Tabular and Tabbing**
`\label`	EqnArray-Equation	**Mathematics**
`\label`	Equation/Display	**Mathematics**
`\label`	SplitEqn-Enter	**Mathematics**
`\label`	Figure	**Floats**
`\label`	Table	**Floats**
`\label`	Ref(within doc)	**Miscellaneous**
`\label`	SectionHeading	**Miscellaneous**
`\labelsep`	Listparameters	**Lists**
`\labelwidth`	Listparameters	**Lists**
`landscape`	ClassOptions	**Preliminaries**
`\large`	TextSize	**Fonts**
`\Large`	TextSize	**Fonts**
`\LARGE`	TextSize	**Fonts**
LaTeX	LaTeX	**Miscellaneous**
LaTeX2ε	LaTeX	**Miscellaneous**
`\LaTeX`	LaTeX	**Miscellaneous**
`latexsym`	PackageNames	**Preliminaries**
`\ldots`	Ellipsis	**Miscellaneous**
`\ldots`	Dots	**Mathematics**
`\left`	BigDelimiters-Left	**Mathematics**
`\lefteqn`	SplitEqn-LHS/OP/RHS	**Mathematics**
`\leftmargin`	Listparameters	**Lists**
`legalpaper`	ClassOptions	**Preliminaries**
`\lengthtest`	Length=Length?	**ifthen package**
`leqno`	ClassOptions	**Preliminaries**
`letter`	DocumentClass	**Preliminaries**
`letter environment`	Letter-DocClass	**Document**
`letterpaper`	ClassOptions	**Preliminaries**
`\lim`	Limits	**Mathematics**
`\line`	Objects	**Pictures**
`\linebreak`	LineBreak	**Miscellaneous**
`\linethickness`	LineThickness	**Pictures**
`\linewidth`	Parameters	**Preliminaries**
`list environment`	List(DIY)	**Lists**
`\listfiles`	Preamble	**Preliminaries**
`\listoffigures`	ContentsList	**Miscellaneous**

\nocite	Ref(other doc)	**Miscellaneous**
\nofiles	Preamble	**Preliminaries**
\noindent	Indent	**Miscellaneous**
\nolinebreak	LineBreak	**Miscellaneous**
\nonfrenchspacing	FrenchSpacing	**Miscellaneous**
\nonumber	EqnArray-Equation	**Mathematics**
\nonumber	SplitEqn-LHS/OP/RHS	**Mathematics**
\nopagebreak	PageBreak	**Miscellaneous**
\normalmarginpar	MarginalNote	**Miscellaneous**
\normalsize	TextSize	**Fonts**
\not	operator $\neg$	**ifthen package**
note	OptFields	**BIBTEX**
note environment	Slides-DocClass	**Document**
notitlepage	ClassOptions	**Preliminaries**
number	OptFields	**BIBTEX**
\numberline	ContentsList	**Miscellaneous**
\oddsidemargin	Parameters	**Preliminaries**
\oint	VariableSizedSymbols	**Mathematics**
onecolumn	ClassOptions	**Preliminaries**
\onecolumn	PageBreak	**Miscellaneous**
oneside	ClassOptions	**Preliminaries**
\onlynotes	Slides-DocClass	**Document**
\onlyslides	Slides-DocClass	**Document**
openany	ClassOptions	**Preliminaries**
openbib	ClassOptions	**Preliminaries**
\opening	Letter-DocClass	**Document**
openright	ClassOptions	**Preliminaries**
\or	operator $\vee$	**ifthen package**
organisation	OptFields	**BIBTEX**
\oval	Objects	**Pictures**
\overbrace	Under/OverBrace	**Mathematics**
overlay environment	Slides-DocClass	**Document**
\overleftarrow	Overarrow	**Mathematics**
\overline	Under/Overline	**Mathematics**
\overrightarrow	Overarrow	**Mathematics**
page	LaTeXCtrs	**Counters**
\pagebreak	PageBreak	**Miscellaneous**
\pagecolor	Color	**Miscellaneous**

`\raggedright`	Page-RaggedEdge	**Miscellaneous**
`\raisebox`	RaiseBox	**Boxes**
`\ref`	Ref(within doc)	**Miscellaneous**
`\reflectbox`	Reflectbox	**Graphics package**
`\refstepcounter`	OpsOn	**Counters**
`\renewcommand`	NewCommand	**Definitions**
`\renewcommand`	PrintStyle	**Counters**
`\renewcommand`	Parameters-OpsOn	**Preliminaries**
`renewenvironment`	NewEnvironment	**Definitions**
`report`	DocumentClass	**Preliminaries**
`\resizebox`	Resizebox	**Graphics package**
`\reversemarginpar`	MarginalNote	**Miscellaneous**
`\right`	BigDelimiters-Right	**Mathematics**
`\rightmargin`	Listparameters	**Lists**
`\roman`	PrintCtr(SpecifyStyle)	**Counters**
`\roman`	SetDefaultPrintStyle	**Counters**
`\Roman`	PrintCtr(SpecifyStyle)	**Counters**
`\Roman`	SetDefaultPrintStyle	**Counters**
`\rotatebox`	Rotatebox	**Graphics package**
`\rule`	Black	**Boxes**
`\rule`	Strut(InvisibleVerticalLine)	**Miscellaneous**
`\savebox`	SaveBox	**Boxes**
`\savebox`	SaveBoxes	**Pictures**
`\sbox`	SaveBox	**Boxes**
`\sbox`	SaveBoxes	**Pictures**
`\scalebox`	Scalebox	**Graphics package**
`school`	OptFields	**BIBTEX**
`\scriptscriptstyle`	MathSize	**Fonts**
`\scriptsize`	TextSize	**Fonts**
`\scriptstyle`	MathSize	**Fonts**
`secnumdepth`	Parameters	**Preliminaries**
`section`	ContentsList-SecTypes	**Miscellaneous**
`section`	LaTeXCtrs	**Counters**
`\section`	SectioningTypes	**Miscellaneous**
`\section*`	see SectionHeading	**Miscellaneous**
`series`	OptFields	**BIBTEX**
`\setboolean`	Boolean	**ifthen package**
`\setcounter`	Parameters-OpsOn	**Preliminaries**

table	LaTeXCtrs	**Counters**
table environment	Table	**Floats**
table* environment	Table	**Floats**
\tableofcontents	ContentsList	**Miscellaneous**
tabular environment	Tabular-Enter	**Tabular and Tabbing**
techreport	DocTypes	**BIBTEX**
TeX	LaTeX	**Miscellaneous**
\TeX	LaTeX	**Miscellaneous**
\textbf	TextStyle	**Fonts**
\textcolor	Color	**Miscellaneous**
\textfloatsep	FloatParameters	**Floats**
\textfraction	FloatParameters	**Floats**
\textheight	Parameters	**Preliminaries**
\textit	TextStyle	**Fonts**
\textmd	TextStyle	**Fonts**
\textrm	TextStyle	**Fonts**
\textsc	TextStyle	**Fonts**
\textsf	TextStyle	**Fonts**
\textsl	TextStyle	**Fonts**
\textstyle	MathSize	**Fonts**
\texttt	TextStyle	**Fonts**
\textup	TextStyle	**Fonts**
\textwidth	Parameters	**Preliminaries**
\thanks	Title	**Miscellaneous**
\the...(counter)	OpsOn	**Counters**
\the...(LaTeX counter)	PrintLaTeXCtrs(DefaultStyle)	**Counters**
thebibliography envir.	Bibliography(DIY)	**Miscellaneous**
\thicklines	LineThickness	**Pictures**
\thinlines	LineThickness	**Pictures**
\thispagestyle	PageStyle	**Preliminaries**
\tilde	Accents	**Mathematics**
\tiny	TextSize	**Fonts**
title	OptFields	**BIBTEX**
\title	Title	**Miscellaneous**
titlepage	ClassOptions	**Preliminaries**
titlepage environment	Title	**Miscellaneous**
tocdepth	Parameters	**Preliminaries**
\today	Letter-DocClass	**Document**
\today	Title	**Miscellaneous**

`volume`	OptFields	**OptFields**
`\vspace`	Vertical	**Spacing**
`\vspace*`	Vertical	**Spacing**
`\whiledo`	while	**ifthen package**
`\widehat`	Accents	**Mathematics**
`\widetilde`	Accents	**Mathematics**
`\width`	BoxDimensions	**Boxes**
`year`	OptFields	**BIBTEX**

Bibliography

[1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeXcompanion*. Addison-Wesley Publ. Co., Reading, Massachusetts, 1994.

[2] David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math*. Springer-Verlag, 1993.

[3] Donald E. Knuth. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1992.

[4] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, Reading, Massachusetts, 1986.

[5] Leslie Lamport. *LaTeX: A Document Preparation System, 2nd Edition*. Addison-Wesley, Reading, Massachusetts, 1994.

[6] Woody Leonhard and Barry Simon. *PC Mom. The Mother of all PC Books*. Addison-Wesley, Reading, Massachusetts, 1994.

Index