MathSpad user manual : draft

Mathpad
User Manual
Draft

Roland Backhouse, Richard Verhoeven and Olaf Weber

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

mathpad@win.tue.nl

January, 1994

Preface

Development of the **Mathpad** 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 Chisholm with technical support from Harm Paas, first became available for use in 1990. The principle 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 and Paul Chisholm left the university world for a post 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 **Mathpad** itself began in March 1992. Richard Verhoeven and Olaf Weber took on the task of a complete reimplementation as final year undergraduate 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 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 **Mathpad**. Ed Voermans suggested the name.

During 1993, **Mathpad** has been used intensively by members of the Mathematics of Program Construction group at Eindhoven University and, once again, the feedback obtained has been extremely valuable in improving the system's design. 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 **Mathpad** was both unstable and sometimes unreliable — Eerke Boiten, Henk Doornbos, Netty van Gasteren, Rik van Geldrop, Paul Hoogendijk, Ed Knapen, Ed Voermans, Harold Weffers and Jaap van der Woude. Lambert Meertens has also offerred sage and much appreciated advice from the very earliest days of the project.

Roland Backhouse
Richard Verhoeven
December, 1993.

Contents

Part I

Introduction

Chapter 1

Welcome

Welcome to the **Mathpad** mathematical-report writer. This introductory chapter has been written to help you assess whether **Mathpad** 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 hard copy of the manual, it is recommended that you not read too far before obtaining a copy of the system and continuing your reading on-line. We promise you 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 install file of the distribution. If the system has already been installed at your site just type the command

```
mathpad manual &
```

and this document will appear on your screen. 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 . (Of course you should not throw the hard copy version of this document away! In many respects a good old-fashioned book is much handier than any computerised system.)

Mathpad 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 flexibility of the TEX and LATEX document -preparation systems[1] by providing a mechanism for converting documents prepared with the aid of the system into LATEX documents. Like these systems **Mathpad** emphasises *logical design* rather *visual design* and is not a WYSIWYG (what you see is what you get) system. Unlike TEX and LATEX , however, **Mathpad** 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.

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

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

or computer programs (with or without mathematical 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
    Variant : m−k}
```

[1] reference to Knuth and Lamport

```
    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 $\lfloor$ 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. **Mathpad** 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.

Mathpad 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:

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

The most important component of the **Mathpad** system is a "stencil"-definition 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 LaTeX output that your notation should generate thus ensuring a high-quality printed version of your document even if the screen version has some shortcomings. (The LaTeX output generated by a user-defined notation is automatically constructed by the system. 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 sometimes required in order to exercise one's right to over-ride the decisions taken by the system.)

The use of **Mathpad** 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 very small in-crowd of researchers you will not have any idea what this calculation is about. That is indeed an advantage since then you will 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.

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

Mathpad 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, **Mathpad** 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 **Mathpad** you will not be sacrificing any of these advantages. (Nor will you have to rewrite all your old documents: a **Mathpad** 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.

Mathpad will not fulfill 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 capabilites for editing and viewing *plain* text are primitive compared to conventional editing systems and prehistoric compared to modern hypertext systems. Even for the input of mathematics its ergonomics is severely hampered by the inhuman nature of keyboard-mouse input. **Mathpad** 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 **Mathpad** then you may continue your reading in the section "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 type in "philosophy" (omitting the inverted commas of course) when the pop-up menu appears. When you have pressed the Enter key on your keyboard the section will replace the contents of this window and you can continue reading. (Alternatively you do not need to type in "philosophy" but can click on View or Load in the pop-up menu that appears — we recommend clicking on View since this gives more protection against mistakes —. This will cause a menu to be displayed containing the names of all the files comprising this manual. If you then select "philosophy" from this menu that section will replace the contents of

this window.)

The next section 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" section replacing the latter everywhere by "tutorial".

If you do not want to leave this section just yet you can bring up the tutorial section alongside this one. Begin by clicking on Edit in the **Mathpad** 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 section by clicking on Done or by using the normal window-manager routine.

Chapter 2

The Mathpad Philosophy

Performing algebraic calculations at a computer terminal and doing them by hand with pencil and paper are quite different sorts of activities. In designing **Mathpad** we started out from the viewpoint that the latter (using pencil and paper) 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, that, for us as human beings, pencil and paper 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.

On the other hand the printed text is a far better medium for communicating the results of mathematical labour to a wide audience than a handwritten document, no matter how beautifully written. The quality of computer-produced mathematical documents has now reached a point that few scientists are willing to do without the assistance of computerised document-preparation systems.

Mathpad is an attempt to bridge the gap between the creative activity of doing mathematical research and the activity of reporting that research to others. We view it as a first-generation, general-purpose, on-screen symbolic calculator. The fundamental principle used in the design of **Mathpad** is that only the simplest and most commonly-occurring syntactic manipulation

primitives should be implemented in **Mathpad** and the user of **Mathpad** should be responsible for the rest.

We do not advocate the use of a computer for algebraic calculations because computers are able to faithfully execute formal manipulation rules. 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 logo that is used for **Mathpad** 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,

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

(Here "$\circ$" denotes composition of functions. The **Mathpad** logo 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,

$$\mathsf{exp}(-x) \;=\; \frac{1}{\mathsf{exp}(x)} \quad \text{and} \quad \mathsf{exp}(x{+}y) \;=\; \mathsf{exp}(x){\cdot}\mathsf{exp}(y)$$

whereas

$$-\mathsf{ln}(x) \;=\; \mathsf{ln}(\frac{1}{x}) \quad \text{and} \quad \mathsf{ln}(x){+}\mathsf{ln}(y) \;=\; \mathsf{ln}(x{\cdot}y) \;\; .$$

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 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. It is such scientists that **Mathpad** aims to cater for.

Chapter 3

Tutorial

Like any sophisticated tool **Mathpad** demands an investment of time and effort in order to get used to it. Moreover, because **Mathpad** 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" oriented to one's own specific needs. It is ironic but true! (The 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 in that much 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.)

The experienced user of **Mathpad** will already have several "stencil files" at their disposal. These files contain specifications of commonly occurring notational elements in the user's documents, ranging from the style for presenting 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 their 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 smaller the delay this 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 **Mathpad** distribution. The **latex** stencil for example, contains the **Mathpad** equivalents of several LaTeX environments, for example the *theorem* environment and the environments for various sorts of lists (*itemize, enumerate, description* etc). After having read through this tutorial you may wish to consult the **Mathpad** document "latex" for a (draft) description of this stencil. (We intend to extend the documentation on the supplied stencils in the course of time. More importantly, we hope that other users of **Mathpad** will make their own (fully documented) stencils available to the user community.)

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 by clicking on Stencil in the **Mathpad** console. A menu will appear and you should select **Gries-Schneider** by clicking on that entry. (Several more stencils have been used to write this tutorial — indeed, all the stencils listed in the menu — 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.)

3.1 The Text Editor

Mathpad 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 **Mathpad**: *expression* and *text*. In addition expressions may either be *displayed* or *in-line*. What you are reading now is of course in text mode. **Mathpad** 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 **Mathpad** console for requests for arguments to commands. In

addition you may consult section 5.4 for full details. This aspect of the system will not be discussed further here.

3.2 Selections

3.2.1 Clicks and Multiclicks

The main goal of **Mathpad** 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. Below a simple mathematical expression has been displayed in order to let you practise the technique.

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

In **Mathpad** there are three types of selection, the target, the source and the parameter selection. If there are three buttons on your mouse these will correspond to the left, middle and right mouse button, 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. (But only because we have set up the stencil in that way. We could have been pernicious and redefined the precedence conventions in any way we liked.) The structure is evident from the spacing around the operators and the parentheses. (**Mathpad** uses a very simple algorithm to automatically insert spaces around operators but if you don't care for the result you can always alter it.) First use the scroll bar to position the expression at the very top of the screen so that you can read and carry out these instructions both at the same time. 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. In each case the subexpression you have selected will be highlighted (in a way 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. Note however

that when you click on either of the two "+" operators you select only the two adjacent arguments. Thus 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 $c{\cdot}d - (e-f){\div}g$. In the case that an operator has been declared to be associative (like "+" 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! For most purposes you will only need the single and double click but sometimes more can be useful.

3.2.2 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 **Mathpad**, so experimentation and perseverance (if necessary) have their rewards. Continue reading only when you feel you have mastered the technique.

3.2.3 Mathpad 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 **Mathpad** is capable of handling various fonts: section headings are in **bold** type, names of **Mathpad** 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 parameter to some function. (Think of emphasising a segment of text. The function is "emphasise" and the segment is the parameter.) How can we distinguish between the text itself and the function applied to the text? Sometimes you might even not wish to see a difference in the final text between the parameter and the function applied to the parameter, 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 parameters 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.

3.2.4 Joining Selections

As mentioned earlier, the three buttons on a mouse are used to make three types of selection in **Mathpad**. (If your mouse does not have three buttons you will need to use the right button in combination with the control key to simulate the third button. You are warned that this feature has never been thoroughly tested.) The left mouse button enables you to make the target selection, the middle button the source selection, and the right button the parameter selection. The final selection technique involves extending an existing selection and 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 will ask you to practise it on text. All that we have so far said and are about to say is nevertheless equally applicable to text and to 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 left mouse 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 right mouse buttons with the pointer positioned between "source" and "selection", and between "parameter" and "selection", respectively. You will see that the source selection is indicated by a thick underline, and the parameter selection by a thin underline.

- target selection
- source selection
- parameter selection

(The hard copy of this manual has been doctored so that what is shown is

what appears on the screen if the instructions have been carried out correctly.)

If you now look at the **Mathpad** console you should be able to locate the "button" labelled Join. Mostly you will only use the target and source selections. Join is, however, one of the few operations that makes use of the parameter selection.

The function of Join is to extend either the target or the source selection so that it "joins" up with the parameter selection. Click on Join with the left mouse button and you will find that the target selection immediately extends so as to include all three items in the list. Set the target selection back to "target selection" before investigating what happens when you click on Join with the middle mouse button. You will find that the source selection is extended to include the last two items in the list. If the relevant selection is not set then Join has no effect. (If you click on Join with the right mouse button you will find that the whole document becomes the parameter selection. This is not very useful we admit. The action should have no effect whatsoever.)

One final point. If you are working simultaneously with two or more **Mathpad** 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.

3.3 Manipulating Subexpressions

The primary goal of **Mathpad** 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, **Mathpad**

provides simple-to-use functions for rearranging expressions, and for finding and replacing mathematical expressions of a given structure. There vare few restrictions on the use of these functions. The principle is that the user knows what he is doing and **Mathpad** is simply there to carry out instructions. There is however one exception, namely, text may not replace a mathematical expression. Thus you cannot perform any operation that would result in text being placed where an expression is expected. For example, you cannot interchange 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 in an expression text may be inserted. Bear this in mind when experimenting with the functions described in this section.

The functions provided by **Mathpad** for expression manipulation are Copy and Swap, Reverse, Distribute and Factorise, Group and Ungroup, Apply , Rename and Find. You will find buttons for each of these functions in the **Mathpad** console. 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 a separate section later. 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.

3.3.1 Target, Source and Parameter Selections

Before proceeding let us recall the three types of selection you can make with **Mathpad**. These are the target selection, the source selection, and the parameter selection. These selections are made using the left, middle and right mouse buttons, respectively. Now is the time to appreciate the meaning of these terms. [1]

[1]It is possible to change the mouse settings around if you wish – for instance if you are left-handed - but this you must do at the level of the window manager. **Mathpad** does

The target selection is where changes to the text occur. With one exception – the **Swap** function which swaps the target and source selections – you may rely on the fact that the effect of **Mathpad** functions is localised to the target selection only. The source selection determines how the target selection will be affected. The parameter selection is typically used to specify the scope of the function that is being applied. You should accustom yourself to which is which: the target selection is indicated on the screen by being displayed in inverse video mode, the source selection is indicated by a thick underscore, and the parameter selection by a thin underscore. 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. You will find, for example, that where the source and parameter selections overlap the area is indicated by two thin underscores. (There are four possibilities: overlapping target and source, overlapping target and parameter, overlapping souce and parameter, 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.)

3.3.2 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.)

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

The expression is incomplete: it contains an expression place holder indicated by a bold face **E**. This is just so that you can practise copying. Set

not provide the facility to change the settings for **Mathpad** alone. Consult your local systems expert.

the target selection to the "**E**" in the expression and the source selection to any subexpression. Then click on the Copy button in the **Mathpad** 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 the Copy button in the **Mathpad** console 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 **Mathpad** into an infinite loop! Note that **Mathpad** 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 associative 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 "·". 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.

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

The overwritten expression is not lost. It is copied to a buffer. The buffer can be opened by clicking on the Buffer button on the **Mathpad** console. The buffer window is just like a normal **Mathpad** edit 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 from the buffer into your edit document. (Remember that the different selections do not have to be in the same window.)

As its name suggests, the Swap button swaps the target and the source selections. (This is the only time that anything other than the target selection is

altered by an edit function.) **Swap** can also be used to swap text or expressions. The two can never be mixed, however, even though there is sometimes no logical reason why not. (This is a flaw in the current implementation.) **Mathpad** also 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 **Mathpad** for this because in the intermediate expression 3^{a+b} most of us would not wish to parenthesise $a+b$. And if $a+b$ is a subscript to 3 as in ${}_{a+b}3$ we again 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 **Mathpad** 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 button labelled "()" in the **Mathpad** console that allows you to insert them. Set the target selection to the subexpression you want to parenthesise and then click on this button. The parentheses will then appear. If **Mathpad** inserts parentheses that you don't want you can remove them with the same button. (Sometimes nothing happens when you do this except that the screen flickers. In that case you can oblige **Mathpad** to remove the parentheses by holding down the "Control" key on your keyboard whilst clicking on the "()" button on the console. 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 section 5.4 for that — but the general message is: you're the boss and can always override **Mathpad**'s actions, and you bear the responsibility.

3.3.3 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, $a = b = c$ then 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 we are 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 **Mathpad** are geared to the use of infix notation in the two ways explained above. When creating a **Mathpad** stencil it is possible to declare that an operator is "associative". This does not mean that the operator is "associative" in the narrow mathematical sense exemplified by addition. It means instead 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" in **Mathpad** 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 "⇒" 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 "associative" operators in **Mathpad**. Our own advice would be to declare all infix operators to be plain "associative" 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 **Mathpad** functions that exploit the notion of associativity are Reverse (called Commute in older versions of the system), Distribute and its inverse Factorise.

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 through 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 on the **Mathpad** console. 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 **Mathpad** reverses as much as possible of the list you have selected.

You may be disappointed that **Mathpad** 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 also be useful in other situations too. One such situation is 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.

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

Working with **Mathpad** it is no hardship to continually 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 with **Mathpad** as follows. First,

factorise "= c" from the expression (using the technique to be explained shortly). This results in

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

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

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

Now with **Find** (to be explained in a subsequent subsection) you can replace all occurrences of "≡" 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 **Mathpad** 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 a (sub)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 argument expression. (In the above example set the target to $a\cdot(b+c+d)$ and the source to $b+c+d$.) Now click on **Distribute** in the **Mathpad** 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 parameter selections are used to specify the function. The two are set so that the parameter 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 parameter selection to b. Now click on **Factorise** in the **Mathpad** console.

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

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. Reset the target and parameter 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** in the **Mathpad** console.

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 in **Mathpad**. The principle is that only the *target* selection

is affected by any **Mathpad** 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 section $a{\cdot}$ selected by the difference between the source and parameter selections has been left unchanged in the process. You can easily return this 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 parameter selections when the latter is a subexpression of the former. If the parameter selection is not a subexpression of the source (including when the parameter is not set) then the source must be a subexpression of the target, and the difference between these two is used as the function.

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 parameter selections, choose the section "$a{\cdot}$" as function to be factorised. Now click on Factorise. One might expect **Mathpad** 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 $a{\cdot}\mathbf{E}$. The subexpression $\mathbf{E}$ is the original target but with all occurrences of "$a\cdot$" removed, whenever they occur.

The design decision underlying this effect is that **Mathpad** functions should, so far as possible, be *total*. No angry beeps, no remonstrances. We, as designers of **Mathpad**., 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:

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

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 **Mathpad** 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

$$\begin{array}{lll} & \{a\ ,\ b\} & \\ \mathbf{O} & \{ & \mathbf{T}\ \} \\ & \mathbf{E} & \end{array}$$

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:

$$\begin{array}{lll} & \{a\ ,\ b\} & \\ = & \{ & \text{set calculus}\ \} \\ & a\ ,\ b & \end{array}$$

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 parameter to "$a\ ,\ b$ " in the *top* line of the calculation, and **Distribute** is activated. (Try it.) The result is:

$$\begin{array}{lll} & \{a\ ,\ b\} & \\ = & \{ & \text{set calculus}\ \} \\ & \{a\}\ ,\ \{b\} & \end{array}$$

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:

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

$$= \quad \{ \quad \text{set calculus} \quad \} \\ \{a\} \cup \{b\}$$

3.4 Stencils

We now come to the most important element of the use of **Mathpad**, the so-called stencils.

A **Mathpad** stencil is a file describing the notational conventions used in one or more **Mathpad** documents, and how these are to be set using the LaTeX document preparation system. 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.

3.4.1 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*Footnote* :Springer-Verlag, 1993. and incorporated them in the stencil "Gries-Schneider". If the stencil is not already visible on your screen you should call it up now. To do so click on the **Stencil** button on your console. A menu will appear with the names of all the stencils used in the preparation of this manual. Select "Gries-Schneider" by pointing at the name and then clicking. (It doesn't matter which mouse button you use.)

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. Much 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.)

Below is an extract from one page of their book (p.151) 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 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: Click on the **Edit** button in the **Mathpad** console and, when a window appears, click on the **Load** button, then the **View** button. Finally, select the entry "GriesSchneiderExample" in the menu that appears.

[2]We assume you are reading this document on screen. If you are reading the hard copy version the only difference with the book is the label number that is given to the property and you will not be able 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.

$(\star y \mid R[x := f.y] : P[x := f.y])$
$=$ ⟨One point rule (8.14)
— Quantification over x has to be introduced. The One-point rule is the *only* rule that can be applied at first.⟩
$(\star y \mid R[x := f.y] : (\star x \mid x = f.y : P))$
$=$ ⟨Nesting (8.20) —Moving dummy x to the outside gets us closer to the final form.⟩
$(\star x, y \mid R[x := f.y] \wedge x = f.y : P)$
O ⟨**T**⟩
E

If you look at the Gries-Schneider stencil you will see that it has six sections, labelled **None**, **Prefix**, **Postfix** , **Associative**, **Left** and **Right**. All but the first of these sections define mathematical operators and their precedences. The first section (the one labelled **None**) 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 six are all associative infix operators, the next two are left-associative infix operators, and the last one 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 *rightmost* 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 **Mathpad** as different "versions" of the same operator. In fact the total list of operators defined in the stencil amounts to sixteen. (Go through each in turn to see which have and which do not have different versions. If an entry does not have alternatives then nothing will happen when you click on it with the right mouse button. If it does a menu will appear. Before you can proceed you must dismiss the menu by pointing at one of the entries, pressing down the right 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 left 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".

Linearly: Begin as before by clicking twice on "$\neg$" in the stencil. Now key in an opening parenthesis. You will find that the closing parenthesis

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

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. (Unfortunately, parentheses are introduced by the system around $p \Leftarrow q$ giving you $\neg((p \Leftarrow q) \wedge \mathbf{E})$. This is not intended — sorry — but you can remove the parentheses by setting the target selection to $p \Leftarrow q$ and clicking on the "()" button on the **Mathpad** console. Then reset the target selection to the expression place holder.) Key in "r". Again, the construction is completed when you press the space bar or "Enter" on your keyboard. (If instead 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 also 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. Specifically, each rule typically has a label — "8.22" in the above example, a name — "**Change of dummy**" in the above example —, a condition or context under which the rule is valid — "Provided $\neg occurs(`y' , `R , P')$ and f has an inverse" in the above example, and the statement of the rule itself — "$(\star x \mid R : P) = (\star y \mid R[x := f.y] : P[x := f.y])$" in the above example. Clicking on "rule" in the stencil with the left mouse button causes the following template to appear in your document at the location of the primary selection:

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

The three bold **T**'s and the bold **E** are place holders for text and a mathematical expression, respectively. In Gries and Schneider's book the headings "**Name**", "Condition:" and "Statement:" do not appear. The template has been so defined that the headings *do* appear on the screen, in order to remind the writer of the function of each of the place holders, but *do not* appear in the LaTeX output generated by the **Mathpad** 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 right mouse button. A menu will appear with three entries "Name, Condition, Statement", "Condition, Statement" and "Name, Statement". If you know press down on any any of the mouse buttons 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. 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 right 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:

(**EV** | **E** : **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 primary selection. Then, with the left 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 **None** section of the stencil is labelled "**E**(**E**)". Clicking on it (in the stencil window) with the left mouse button generates the template **E**(**E**) at the position of the cursor in your **Mathpad** document. This template has been used to construct the expression $occurs(`y' , `R , P')$. Specifically, the first place holder has been replaced by the expression *occurs* and the second place holder by the expression $`y' , `R , P'$.

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

The postfix operator **[V := 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 definition 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 indentifier 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 "**E(E)**" will never be parenthesised by **Mathpad**. The point is that the automatic insertion of parentheses is applicable only to the templates defined outwith the **None** section. **Mathpad** 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 that has 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 "()" button on the **Mathpad** console.

Instead of defining a postfix operator **[V := E]** we could have defined a postfix operator **[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. **Mathpad** 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 "**O**⟨**T**⟩". This is an excellent example of the flexibility of the template notion in **Mathpad**. 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 operators 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 click on Disp in the **Mathpad** console. A bold-faced **D** will appear in the text like the one below. The target selection will be set to the "**D**".

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 **O**⟨**T**⟩ in the Gries-Schneider stencil. This is what you should get.

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

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 **O**⟨**T**⟩ behaves exactly like an associative 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.

3.4.2 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 **Mathpad**. 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 **Mathpad** is not adequate to your needs. In this section we only outline the creation of stencils and you are referred to the reference section for more details.

To create a new stencil click on Stencil in the **Mathpad** console. Respond to the menu by choosing New Window. 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 **Mathpad** 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 ("Prec.", "Space" and "Kind") must always be set. When you open the window these items are already initialised to the default values 0,0 and "none".

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 **Mathpad** will try to create one for you. All the names in the Gries-Schneider stencil were created by **Mathpad** but for "rule", "Quant" and "mul".) Now select the menu item "Prec." and choose a number from the menu that appears. After having done so select "Kind" and choose an item from the menu that its selection causes to appear (for example "Associative"). Now move the mouse pointer down into the main window area and click. The three headings "Name", "Screen" and "Latex" 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 **Mathpad** 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 instance of the (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 **Mathpad** is in automatic generation of LaTeX mode.) 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 the "Latex" field. This process is then repeated for $\oplus$ and $\bigcirc$. 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 alignment of different elements, for example, 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 blueprint. 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 definition will be copied across, including the LaTeX code.

Using the 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.

3.4.3 Find and Replace

The **Find** (and Replace) function is a very powerful and easy-to-use facility. If you click on **Find** in the **Mathpad** console a window will appear divided into two equal-sized subwindows. The top one specifies what is to be found, and the bottom one what the replacement should be.

The important point to remember about these two windows is that they are just like any other **Mathpad** edit window. Everything we have explained above about editing expressions can be carried out in these windows. 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 another place holder then the number is copied too, otherwise **Mathpad** creates a new number. For instance it is possible to create the expression **E1−E1** in the find window. If you create the expression in the normal way **Mathpad** will create **E1−E2** . Now you can set the target to **E2** and the source to **E1** and copy the latter to the former thus creating the desired expression. Searching for **E1−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 **E1−E1** to 0, is not so useful since it is so much quicker and easier to do by more primitive means.) At present it is not possible to maintain a database of such transformations in **Mathpad** — see the preface for reasons why not — but the option is being reconsidered.

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

There is one catch to the use of find-and-replace that may cause you some bother. Once you understand its cause you will not be bothered any more.

The catch is that before initiating a find or find-and-replace you must always reset the target selection in the document you are editing. The explanation is simple. Remember that the find and replace windows are just like any other **Mathpad** document. Remember also that you may be simultaneously editing any number of documents but that each selection, and in particular the target selection, is set in only one document. Well, once you have prepared your find-and-replace action the target will inevitably be in either the find or the replace window, and more than likely at the end of one of those windows. Initiating find (or find and replace) will thus cause **Mathpad** to search in that window and not in the document you are editing. If the message "Tree not found" appears in the console then this could be the reason.

A very pleasant aspect of **Mathpad**'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.

3.5 Conclusion

This completes the tutorial. Best of luck with your further efforts at using **Mathpad**. We'd be pleased to hear of your experiences. And remember. Experiment as much as you like!

Part II

Reference Manual

Chapter 4

The Basics

Mathpad works in the X11 window environment and we assume that the user is familiar with the basics of this environment. This section introduces basic terminology used throughout the manual and describes briefly each type of **Mathpad** window.

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

Mathpad uses three mouse buttons. If your mouse has fewer buttons, **Mathpad** could be difficult to use. More buttons is usually no problem. The first two buttons are used only for making a selection or positioning the cursor. The third button is used to obtain a popup menu or to make a special selection.

The click and a drag can be used in **Mathpad**. A click is a fast press and release of a mouse button. A drag is the sequence: depress a mouse button, move the mouse and then release the mouse button. Also the double click can be used, which is a click immediately following a click or drag. The double drag is defined the same way. A sequence of clicks and drags is made

using the double click repeatedly, which results in triple clicks, triple drags, quadruple clicks, quadruple drags and so on.

Mathpad 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 press down the command key, then press and release the appropriate alpha-numeric key. In this manual, use of the three command keys is 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 your keyboard). There is no difference between **C-u** and **C-U**, for example, both being equally acceptable.

The key used for the meta key is one of the modifier keys from 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.

During a session there is one *target selection* in the editor, where the keyboard input is entered. The term selected by the target selection is called the *target term*. The target selection is displayed in reverse mode. Several functions use the target selection or target term as an argument. There is also a *source selection* and a *source term* which are used by functions with two arguments. This selection is underlined with a thick line. For the functions distribute and factorise a *parameter selection* is available together with the *parameter*, which is underlined with a thin line. These three selections can be made with the mouse, where the leftmost button is used for the target selection, the middle button is used for the source selection and the rightmost button for the parameter selection. (If you wish to change these settings, for instance if you are left-handed, you should use the facilities in X11 for that purpose.) From now on, whenever we refer to **Mathpad**-specific commands we use the terminology *target*, *source* and *parameter*. When we refer to X11-specific commands we use the terms *first*, *second* and *third* mouse buttons. If no mention is made of a specific mouse button then which button is

used in not significant.

Underlining ensures readability on a black-white monitor. Support for colour monitors will be added in a later release.

The following font styles are used to highlight different items.

Bold Keyboard commands or a sequence of keyboard commands.

Italic Keywords and important items. *Bold italic* indicates an indexed item.

Sans serif Options on different menus. These options can be buttons at the top of a window or an item in a popup menu.

`Type writer`
LaTeX macros.

Buttons

In the different windows there are several *buttons* which are used to execute a specific command. A button looks like a box with a command in it. You can execute the command by clicking in the button. By dragging outside the button, a selection is undone and the command will not be executed. Some buttons have a popup menu attached to them. This popup will appear when the right mouse button is used to click on the button.

Scroll bars

In some windows there are scroll bars to move the main contents of these windows around. A scroll bar contains two arrows and a black bar which indicates the relationship of the displayed information to the total document. The black bar can be moved by clicking or dragging in the scroll bar. By clicking on an arrow the bar moves in the direction the arrow is pointing. Otherwise, the bar is moved according to the mouse button that is used. A click with the first button moves the bar down or left, a click with the second

button moves the bar to the position of the mouse cursor and a click with the third button moves the bar up or right. It is also possible to drag in the scroll bar, where every move is handled as a click.

Pop-Up Menus

When a filename is needed or a serious error occurs a popup menu will appear. A popup will block the program and the normal edit actions can not be used. When the popup is removed the editing can continue. Some popups are shown in figure 4.1 and how the popups can be used will be discussed in section 4.1.

Mathpad Console

When the editor is started, the user is presented with the interface shown in figure 5.1. We call this window the **Mathpad** console. In the console two rows of buttons are available to open other windows or to apply functions to the selections. There is also a small strip below the buttons where messages are displayed. The functions associated with each button are discussed in section 5.2.

Edit Window

When the edit button in the main menu is clicked, an *edit window* as shown in figure 5.2 comes available. When the window is opened the contents will be empty. Each edit window contains a document and the name of that document is shown in the header of the window. A user can open several edit windows so that different documents can be combined or parts can be copied to other documents. An edit window can also be used as a scratch pad to hold temporary structures and parts. More information about the edit window can be found in section 5.3.

The Buffer

The *buffer* is used by the system to record parts of a document that have been deleted as a result of some edit command. It is just like any other **Mathpad** edit window except that its contents cannot be saved; it may thus be usefully used for scratch work. It is opened by clicking in the buffer button on the **Mathpad** console. Full details of its use can be found in section 5.10.

Find & Replace Window

When something must be changed a number of times, the *find-and-replace window* can be used. This window contains two parts both of which can be used like a normal **Mathpad** edit 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 section 8.

Symbol Palette

Many documents contain special symbols, especially mathematical documents and text in foreign languages. To make these symbols easily accessible to the user, the *symbol palette* is available. It can be opened by clicking in the symbol button of the main menu. This palette displays a subset of the available symbols, as shown in figure 7.1. Via a palette the symbols can be used. More information can be found in section 7.1.

Stencil Palette

To use defined stencils the *stencil palette* is invoked. Opening a stencil is initiated by clicking on the stencil button on the console. Figure 7.2 shows an example of a stencil. It is possible to open several stencils. Each stencil consists of a number of templates each of which may have several versions. The use of stencils is discussed in section 7.2.

Define Window

To make the editor usable for all kinds of applications the user can define his own stencils. This is done in the *define window*, shown in figure 9.1. The define window can be opened by clicking in the define button of an opened stencil. Creating and updating stencils is described in section 9.

Defaults Window

In the *defaults window*, as shown in figure 10.1 some default settings of the editor can be changed. These settings can be used to change the layout on screen and LaTeX or to change the directory names the editor uses. Exactly how to change the settings is explained in section 10.

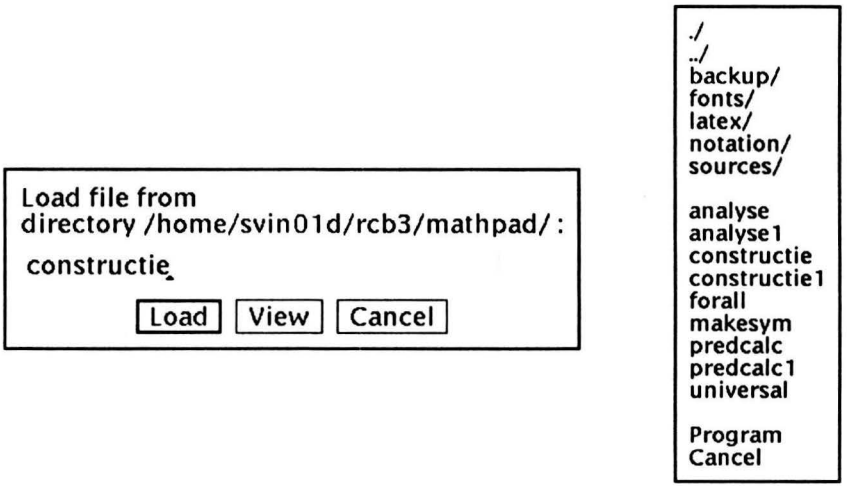


Figure 4.1: Some pop-up windows

4.1 Pop-Up Menus

The *pop-up menus*, as shown in figure 4.1 are used to ask for a name or to report an important message. While a pop-up is displayed, the editor will ignore all actions until a button of the pop-up is used or a line is selected.

The left pop-up in figure 4.1 is used when a document is loaded. Such a pop-up will also appear when documents or stencils are saved, included or renamed. The filename can be entered and the related action will be performed by clicking on one of the buttons or by pressing the return key which is the same as clicking on the first button. In this case the file will be loaded from the directory that is displayed in the window or from the program directory if it is not found. When the entered file is a directory, the contents of the directory will be shown in a pop-up menu that looks like the pop-up on the right side.

The right pop-up can be used to make a selection from a number of options. Each line can be selected and only one selection can be made. In this pop-up the contents of a directory is shown where first the subdirectories are listed, then the files and finally two options. When a subdirectory is selected a pop-up will appear with the contents of that subdirectory. When a file is

selected it will be loaded. The two final options are available to view the contents of the program directory or to cancel the action. Such a pop-up will only appear when a document or stencil file is loaded.

There are also pop-ups which are used for menus. These menus contain options which are available at that moment and can be popped up by clicking with the third button at the position where a menu is available. The click will position the menu at the position of the cursor and an item can be selected by clicking on it. The drag will show the menu and the selection must be made immediately. At the moment menus are available for the buttons **stencil** and **version**, in the stencil window, at the top of the stencil define window and in the default window where fonts can be entered. The menu for the stencil button contains a list of loaded stencil files. The menu for the version button and the stencil window contains a list of possible versions for the selected stencil. If the list is empty, the default action will be executed and no pop-up will appear. The pop-up in the default window contains a list of possible fonts that can be used. Defining templates in a stencil is menu-driven; selecting any of the items displayed at the head of a stencil define window will cause a pop-up menu to appear.

The pop-ups are also used to report serious errors. Such a pop-up can be removed by clicking in the pop-up.

Chapter 5

Getting started.

This section describes how you start with mathpad and how you carry on. The subsections describe a number of the windows, how text and expressions are edited and what keyboard commands are available.

5.1 Starting Mathpad

Mathpad is started with the command `mathpad`. Unlike other editors, **Mathpad** does not use the directory in which it's started. Instead, a default work directory is used which is specified in the file `~/.mpdefaults`. That file contains the directories where the documents, stencils and LaTeX files are found and written by default.

If the file with the defaults does not exist or the specified directories are not available, **Mathpad** will create these directories on demand. By default, **Mathpad** uses the directories `~/mathpad`, `~/mathpad/stencil` and `~/mathpad/latex` for the different files. These defaults can be changed in the default window which is opened by clicking on the **Default** button of the menu window.

After **Mathpad** is started, the console will appear. The console contains

a number of buttons to open other windows or to use functions. The Edit button is used to open an edit window which is needed to edit a document.

Options.

A number of options can be used when **Mathpad** is started. The possible options are:

-display *name*
: Specifies the name of the terminal where the windows should be displayed. On this terminal the X window environment must be available. The *name* usually looks like name.of.terminal:0.0.

-iconic The windows will start as an icon. This can be used to start **Mathpad** in startup script.

-ascii Generate ascii instead of LaTeX . By default, **Mathpad** produces LaTeX code and a number of specific rules are used to generate correct expressions. This is visible in the define window where the LaTeX is shown. With this option, the ascii will be shown and the rules are not used. LaTeX can still be produced as a second option.

-plain Produce LaTeX code almost as you would type it. **Mathpad** will not add the spacing around operators, which is now done by the LaTeX program itself.

-project *name*
: Start **Mathpad** with the configuration as described in the project file *name*. A project file contains information about the defaults, the opened windows and the loaded documents and stencils. The defaults from a project file overrule the defaults from the default file. A project file is made during an earlier session with **Mathpad**. If *name* does not start with a dash (-), the -project part can be omitted.

If you use an argument beginning with a dash (-) which is not listed here, **Mathpad** will use that argument as a keyboard definition option. For example, if you use the argument -emacs, mathpad will load the emacs key-

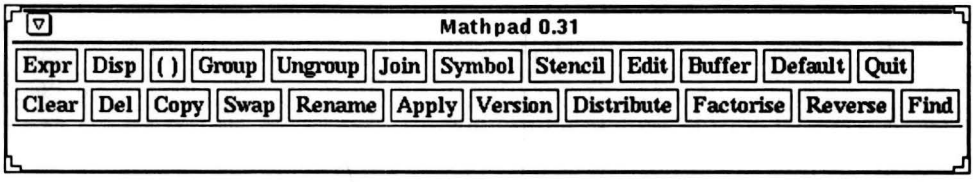


Figure 5.1: The Mathpad Console

board definition (the only one available). If the definition file does not exist, **Mathpad** will not be very useful (keyboard input will be disabled).

It is not possible to start **Mathpad** with a normal ascii file or document as argument.

5.2 The Mathpad Console

The **Mathpad** console, which appears when mathpad is started, shown in figure 5.1, is used to open the different windows and palettes that mathpad uses. Also a number of functions can be called via the available buttons. These functions usually applied to the selections, which can be in different windows. The meaning of each of the buttons is:

Expr	Insert a place holder for a mathematical expression.
Disp	Insert a place holder for a displayed mathematical expression.
( )	Put parentheses around an expression or remove them.
Group	Make the target term a proper subexpression of the expression in which it occurs.
Ungroup	Insert the selected subexpression in the expression one level higher.
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.

Symbol Open a window which allows the use of special symbols.

Stencil Open a window which allows the use and creation of stencils and operators. A click with the third button will show a list of already loaded sets of stencils.

Edit Open a window to begin on a new document. A user can work on different documents at the same time.

Buffer Open or close the window to restore parts of documents that have been deleted.

Defaults Open or close the window allowing adjustment of system defaults.

Quit Remove all documents that do not need to be saved and exit the system if there are no documents left.

Clear Unset the selection indicated by the used mouse button.

Del Erase the selection and put it in the buffer.

Copy Copy the source term to the target term. The target term can be restored via the buffer.

Swap Interchange the target and source terms in the document.

Rename Replace every occurrence of the target term by the new value in the source term. Both the target and source term should be identifiers. The range of the replacement is selected with the parameter selection.

Apply Apply the function selected by source term and parameter to the expression in the target term. For example, apply '$a+$' to '$b \times c$' giving '$a + b \times c$'.

Version Replace the stencil in the target term by the next version of that stencil. A click with the third button will show the list of available versions.

Distribute Distribute the function selected by the source term and parameter over the expressions in the target term.

Factorise Remove the function selected by the source term and parameter from all expressions in the target term. This is the inverse of distribute.

Reverse Reverse the arguments of the expression or stencil in the target selection.

Find Open or close the window to find and replace stencils or large constructions.

Below the buttons there is a small strip where the messages from the system are displayed. It is also used for entering the arguments for the find and query replace functions.

5.3 The Edit Window.

To edit a document an edit window, as shown in figure 5.2, must be open. If there is no edit window available, one can be opened by clicking in the **edit** button of the main menu. It is possible to open multiple edit windows to be able to combine documents or to simultaneously edit documents.

The Buttons

At the top of the window some buttons are available to start or end a given document. The following functions are related to the buttons

Load Open a pop-up window to enter the name of document that must be loaded. The related file will be searched for in the directory of the user and will be loaded if it is found. When a document is loaded, the old document in the window is removed and saved in a backup file. The stencils that were loaded when the document was saved will be loaded to be able to convert the document to LaTeX . Also ASCII files can be loaded, although they will be filtered to

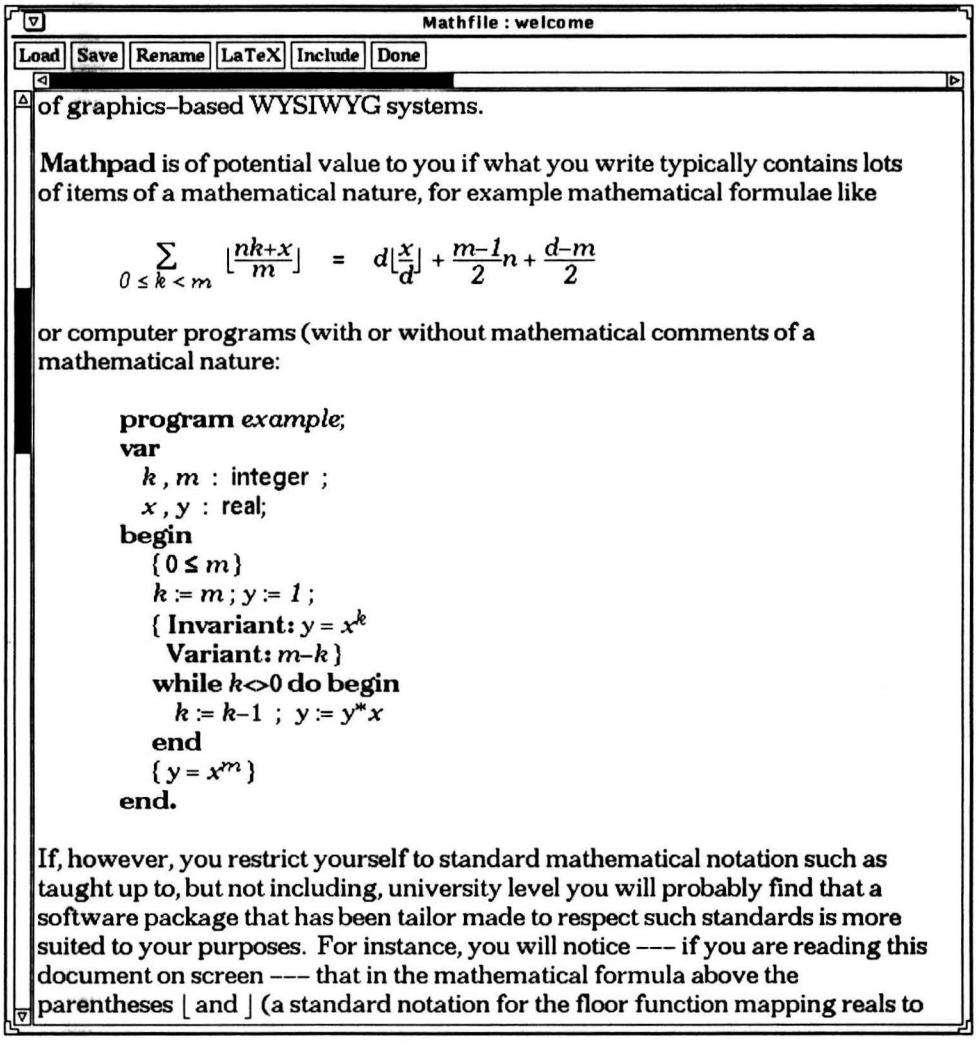


Figure 5.2: The edit window

remove all the strange symbols. The name of the window and the icon is changed to the name of the file.

When the View button from the popup is used the stencil files are not loaded. The file can be edited like a normal document although the changes will not be saved during a periodic save operation.

Save Save the document that has been edited. The name of the document can be changed in the pop-up window that will appear. The document will be saved in the directory of the user and a message will appear if the document can not be saved.

Rename Rename the document. The name of the edit window and the related icon will be changed but the file will not be saved.

Latex Translate the document to LaTeX . The name of the LaTeX file will be the name of the document with the extension '.tex' but that name can be changed via the pop-up window that appears. The file will be placed in the LaTeX directory of the user. It is also possible to convert the document to ASCII by using the Ascii button. This ASCII mode will be extended in the future to enable interfaces with other packages.

Include Include a document at a position of the cursor. The document can be a document made by **Mathpad** or a normal ASCII file. To include a file successfully the cursor must be in the text.

Done Close the edit window. If the document has not been saved a popup window will appear and the document can be saved or the action can be cancelled. Otherwise the document will be removed without a message.

The name of the edit window and its icon will be the name of the file without the specified path. When an edit window is opened, the name of the document will be noname?? where ?? will be a number that indicates how many edit windows have been open. The name can be easily changed when a new document is made and will change when a document is loaded. To start a new document a new edit window must be opened.

It is not yet possible to open multiple edit windows for the same document. Every edit window will contain a copy of the document and edit actions in

one edit window will only change that copy.

5.4 Editing commands

Plain (non-mathematical) text can be edited with a number of keyboard commands. The commands are based on the emacs commands and most of the usual functions are available. Some of the functions behave a little differently compared to normal editors. This is due to the possibility to use proportial fonts and to insert expressions, which are handled as one symbol.

5.4.1 Cursor movements

The following commands move the cursor or the target selection.

C-F Right Moves forward one character

C-B Left Moves backward one character

M-F Moves forward one word

M-B Moves backward one word

C-N Down
Moves to the beginning of the next line

C-P Up Moves to the beginning of the previous line

C-E End Moves to the end of the current line

C-A Home
Moves to the beginning of the current line

M-< Moves to the beginning of the document

M-> Moves to the end of the document

M-R Moves to the center line of the window

5.4.2 Scrolling the window

There are some commands to scroll the contents of the window. The cursor or target selection will not be changed.

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.

Unlike the emacs commands, **C-V** and **M-V** will not change the position of the cursor or target selection. The combination **M-R** can be used to move the cursor to the center line of the window. **C-L** can be used to go back to the position of the cursor.

5.4.3 Erasing and retrieving text

For erasing and retrieving text the following commands can be used.

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

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

C-K Kill the target selection if available. Otherwise all the characters to the right of the cursor up to the next newline or expression will be killed. If the cursor is at a newline or expression, it will be killed.

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

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

C-W Kill the target selection.

CM-W Append the text which will be killed after this command to the already killed text.

C-Y Retrieves the most recently killed text and insert is 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 and a new kill command will place the text in the buffer where it can be restored later.

Transposing text

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

C-T Transposes the characters to the left and right of the cursor and moves the cursor one position to the right. Many typing errors can be recovered using this command.

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 situation as a single character which is not part of a word. These commands are not available to transpose parts of one expression.

5.4.4 Find and replace

The emacs functions incremental search and query replace are also available. These commands can be called with the following key combinations

C-S Start the incremental search. Mathpad will ask for a string in the menu window and will search for the string as it is being entered. The target selection or cursor will be used as starting position. During the search the following keys or actions can be used:

a template
: If the search string is empty the template will be used as search string, in which case it cannot be edited. Otherwise the template will not be used.

any printable character or symbol
: Add the character to the search string and search for the next occurrence.

C-S Search for the next occurrence of the search string. If the last search action failed mathpad will start searching from the beginning of the document. If the search string is empty the last string will be used.

C-R Search for the previous occurrence of the search string. If the last search action failed mathpad will start searching backward from the end of the document. If the search string is empty the last string will be used.

ESC End the incremental search and leave the selection at the current position.

DEL Move to the previous position that was found.

C-G Cancel the incremental search and move the selection back to its original position.

Note that the incremental search acts like the same function from emacs, although not all features from emacs are available.

C-R Start a backward incremental search. The commands that can be used are the same as the commands fora normal incremental search (**C-S**).

M-% Start a queried replace action. The scope of the query-replace will be the source selection if it is placed in the same window. Otherwise the scope will be the part of the document from the target selection or cursor till the end of the document. The query replace will ask for two strings, the source string and the

replace string. The strings can contain normal characters and symbols and is ended by a return. Also stencils can be replaced by selecting one stencil as source string and another stencil as replace string. The replacement of stencils will also replace the information concerning the kind of stencil, spacing and precedence. Missing arguments of the operator are not added. After the source string and replace string are entered the editor will search for occurrences of the source string within the scope and will prompt for each found occurrence if it must be replaced. The possible responses to this query are:

y, Space
: Perform the replacement.

n, DEL Skip this occurrence.

! Replace all remaining occurrences without asking and move the cursor to the last replacement.

q, ESC Stop the query replace.

C-G Stop the query replace and move the cursor back to its old position. Replacements already carried outare not undone.

The incremental search and query replace are also ended by clicking in an edit window.

5.4.5 Arguments to commands

Some commands that can be used can also be called with an argument. By default this argument is 1 but it can be changed using the **C-U** key. After using the **C-U** key a number can be entered which may not be negative. The argument can be ended using the **C-G** key without applying a command. Otherwise the command that is used after the last digit will be performed repeatedly. For example, **C-U 2 0 A** will insert twenty A's. There are some commands that do not use the argument and they will perform the standard functions.

5.4.6 Fine tuning the tree spacing

The space around infix operators is calculated by the editor and will not always be perfect. To be able to change the spacing, the following commands can be used:

C-+, C-= Increases the spacing around the operator.

C-- Decreases the spacing around the operator.

C-0 Resets the spacing around the operator to the spacing calculated by the program.

The adjustments to the spacing will be added to the calculated spacing, which can change if the tree is edited. The fine tuning can best be done when the expression or document is almost finished.

5.4.7 Short cuts for buttons

At the moment the following short-cuts are available for the buttons from the main menu.

CM-U Ungroups the selected expression.

CM-G Groups the selected expression.

CM-C,F9 Reverses the sub-expressions and operators in the target selection. If the target selection has no sub-expressions or operators, the selection will be swapped with the structure of the same kind on the left side of the selection. If such a structure does not exist on the left side, the right side will be be used.

CM-F Factorises according to the indicated selections.

CM-D Distributes according to the indicated selections.

F4 Removes the (outermost) parentheses from the target term.

F5 Inserts an expression place holder at the position of the cursor.

F6 Inserts a place holder for a displayed mathematical expression at the position of the cursor. The expression will be typeset in the modified tabbing environment.

F7 Copy the source selection to the cursor or target selection.

F8 Swap the target and source selection.

5.4.8 Miscellaneous

The following commands are available to make working with mathpad a little easier.

M-L Moves to the line that produces the *n*th LaTeX line. Mathpad will ask for the number *n* in the menu window.

M-P Save the current window positions in a project file. The file will contain the positions of the edit, stencil and symbol windows and which file or page they contain. The contents of the windows is not saved. The name of the file can be entered at the prompt in the menu window and will be placed in the mathpad directory of the user. A project file can be used at the command line when mathpad is started, like '**mathpad** *name*'. The project file will get the extension '.mpj'.

CM-E, CM-O, CM-I, CM-V, CM-T
Inserts a place holder with the related symbol.

C-. Remove the dots around a text place holder or 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 **Disp** button from the menu window or the **F6** key, to the left or right by one tab distance. When all displayed expressions should start

at the same distance from the left margin, it could be that some expressions need an extra indentation, while others need less indentation.

F3, F18 Paste the selection from the window system at the position of the cursor. This selection can only be text. On a sun keyboard the F18 key is labeled with Paste.

F10 Convert the target selection to LaTeX . The selection will be available through the window manager and can be pasted into another window, for example a text editor. At the moment, the LaTeX can have 3 different versions. Depending on the argument, constructed with **C-U**, the output will be:

- **1** the default LaTeX produced by mathpad.
- **2** the default LaTeX where **\ms** is replaced by a number of **\,** commands.
- **3** the default LaTeX with no **\ms** commands and no braces **{}** around the operators.

Normal text without any special symbols or stencils will be converted to ASCII, so normal text can be selected to paste it in other windows.

The keys F10 and F18 can be used to check the LaTeX code produced for a particular expression or part of text. This is the reverse of the previewing which is normally done when a text editor is used.

Furthermore some normal keys will get a special treatment when they are used in expressions. The **space** character can not be inserted in an expression or identifier and when this is tried, the editor will move the selection to the next empty place holder or text position, whatever comes first. If the text position is found, the space will be inserted at that position and not inside the expression. 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 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 **return** key has the same behaviour except that the return will not be inserted in the text. Also the **C-return** key can be used for that action with the extension that it will also end a text place holder.

The (key will place braces around an expression or insert it in an identifier. The) key will insert a brace in an identifier if the matching opening brace is in that identifier. Otherwise the smallest embraced 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.

The **CM-** combination can be used to check what combination mathpad receives for a particular key combination. The **C-A** must be used to stop this feature, which is only useful for testing the keyboard when a keyboard definition file is made.

5.5 Editing Text

An edit window can be used like a normal text editor. The text is entered with the normal keyboard keys and the cursor keys can be used in the usual way. A kill buffer is available which can be used to move, remove or copy parts of text. Characters and words can be transposed and a find and replace function can be used.

The primary selection can be used for selecting text, and selections can be exported through the window system. The selected text is translated to LaTeX to make sure 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 mathpad document in another window are:

- Select the text (or expression) that should be converted to LaTeX with the primary selection.

- Press the **F10** key. The selection is now converted to LaTeX and will be available until something is selected in another program. Using the selection in mathpad is still possible.
- Paste the selection in the other program, which is done by using the correct key, mouse button or menu option.

The selection from another program can also be pasted in mathpad. The selected text will be inserted at the position of the cursor. The steps to copy text from another program to mathpad 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.
- Paste the selection in mathpad by using the **F3** or **F18** key. The last key is labeled with Paste on a Sun keyboard.

In mathpad, special characters can be inserted in the normal text. These symbols are selected from a symbol palette and are handled like normal characters in the editing actions. However, if a text contains such symbols, the LaTeX code will be a little different. The symbols are bounded to LaTeX macros and that macro will be used in the LaTeX code. The dollar signs (**$**) are automatically added to switch to the correct mode and are therefor not needed in the normal text.

Templates can also be used in text. These templates can contain text place holders and if such a template is used while a part of text is selected, the selected text will by used at the first available position. This makes is possible to add font and size changes to parts of text after the text is entered. Just select a part of text and click on the correct template. When the LaTeX code is generated the templates determine which LaTeX code should be produced.

A text place holder can be edited like normal text and can contain the same items as normal text, such as expressions and special symbols. To start with the text in a place holder, click on it and just start typing. In some situations, the place holder is used inside the tabbing environment, which disables the

line breaking routines from LaTeX . The line breaks should now be inserted by the user and the proportianally-spaced font helps in this situation.

At the moment, there is no automatic line wrap and there are no features to fill out a part of text. These features may be added in a future release.

5.6 Entering Expressions

To enter an expression in the text, an expression place holder is needed. The different ways to get such a place holder are:

- Click on the **expr** button of the menu window or use the **CM-E** keyboard shortcut. The inserted expression will be inline and a template or identifier can be entered.
- Click on the **disp** button of the menu window or use the **CM-P** keyboard shortcut. The displayed expression is set in a tabbing environment, so templates which use tabbing commands can be used now.
- Click on a template from a stencil palette. The template will be inserted and the cursor will move to the first place holder in that template. If that place holder is not available, the cursor is placed behind the template.

After inserting an expression, the empty place holder is selected. The place holder can now be filled with the desired expression. The expression can be entered in different ways:

top-down An expression usually contains a number of templates and operators at different levels. The top-down approach first enters the main operators from the highest level. The open place holders are now 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 make a subexpression first and then add the operator from a higher level. In the bottom-up approach, the leftmost subexpression is entered first. The operators are entered later and the selected expression is used as the first argument of the operator. The problem with the bottom-up approach is the motion of the mouse. After finishing a subexpression, it should be selected with the mouse, which is also needed to add the next operator.

left-right
: The third possibility is entering the expression from left to right. This is very natural because you also read that way. At the moment the keyboard shortcuts for templates are not fully supported, so you still have to switch between the keyboard and the mouse. While entering the expression, the parentheses and precedences are used to add operators in a context sensitive way.

The difference between these three approaches is now shown with an easy example from boolean calculus: $\neg a \wedge \neg b \equiv \neg(a \vee b)$. After inserting an expression place holder, the following steps are needed for very approach.

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

In this table, '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 first selection.

The number of steps does is not the only information this table shows. 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' 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. The top-down approach and the left-right approach are in this situation almost equal. The bottom-up approach is clearly more expensive.

Each approach uses some of the features which make the editing easier. These features are:

- The return and space will move to the next empty place holder from the current expression. The top-down approach uses this after every identifier.
- A selected expression will be used as the first argument of a template or operator. An existing expression can therefor be used as a subexpression. This is used in the bottom-up approach.
- When the cursor is still *in* the identifier and an operator is used, the left argument of the operator will be the subexpression which contains only operators with higher precedence and is completely at the left of the cursor. Parentheses can be used to make sure that only the expression within the parentheses will use as subexpression. The left-right approach uses this very often.
- 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-right approach in order to enter $\neg(a \vee b)$ in stead of $\neg a \vee b$.
- It is possible to make a sequence of operators like $a \wedge b \wedge c$, which is done by selecting one part of available expression and use the same operator again. This can be used in all approaches. In the top-bottom, you just click twice on $\wedge$. In the bottom-up approach, select $a \wedge b$ or b

and click on $\wedge$. In the left-right approach, click on $\wedge$ after typing the b.

It is possible to use the approaches and features in a mixture. Depending on the situation, one of these features can be used.

5.7 Using stencils

To make an expression in an easy way templates should be available in a stencil palette which will be discussed in section 7.2. A stencil palette can be opened by clicking on the **stencil** button of the menu window. The stencil can then be loaded by clicking on the **load** button of the stencil palette. The loaded templates can be selected with the mouse by clicking on them and the template will be used on the position of the *target selection* if possible. The template can be used to replace an expression, operator or text and the type of the selection will determine how the template will be used. If the template can not be used, it will be ignored. Some examples:

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

As evident in the examples the selection will be used as the first correct argument of the selected template or the arguments that the operator needs. If the target selection is empty, the template will be inserted at the position of the cursor, as shown in the sixth example. The arguments of an operator are automatically added unless the selected position is already an operator.

After using a template the selection will be set to the first empty argument of the expression. If there are no empty arguments, the selection will be

placed behind the expression. The selection will therefore be at the most likely position where the next expression or text is entered.

When a part of text is selected, a template with a text place holder can be used. The text will be inserted in the first text place holder, so templates can be used to add font or size changes very quickly. Also an index can be made by using templates. Just select the word and use the correct template. If the template does not contain a text place holder, the text will not be removed and the template will be used just after the selection.

The reverse action is more difficult. A complete template is removed by using the delete key or the **Del** button from the menu window. The copy operation can be used to copy the original selection at the position of the template, which will remove the used template. This copy action is usually straightforward, except when text should be copied. Text can't be copied onto an expression or template. If it does not work right away, remove the template completely and restore the text from the buffer. An other possible solution is killing the text with the **C-K** command, remove the template and restore the text with the **C-Y** command.

5.8 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) left button for the target, middle button for the source, and right button for the parameter selection — and all these selections are made in the same way.

5.8.1 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, which is started and ended with a return. Templates can contain returns also and a selected line can 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 selection will be made.

5.8.2 Expressions.

When an expression is being 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.

In the editor *operator sequences* can be built. An operator sequence is a sequence of expressions over an infix operator like the sequence

$$a + b + c$$

The subsequences are also operator sequences and they can be selected by dragging from the first expression to the last expression of the sequence. The selection will be shown during the drag and it will show that also a sectioned sequence can be selected. The selection, however, will depend on the kind of operator that is used. If the operator is left associative all expressions from the start of the sequence to the pointed expression will be selected. If the operator is right associative all expressions from the pointed expression to the end of the sequence will be selected. Otherwise the selection will be the smallest sectioned operator sequence that contains the two pointed positions. Some possible selections from the sequence $P \vee Q \vee R$ are

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

The double click can be used to selected the complete sequence 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 follows:

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 $\vee$	$\vee j > 1$

The double click can also be used to select templates which 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 which only change the font style or the size of the text.

5.8.3 Large selections.

Large selections which don't fit on one screen can be made using the **Join** button from the menu window. This function expands the selection to contain the *parameter selection*. The mouse button indicates which selection should be expanded. The parameter selection will expand to contain the source selection. So, 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 parameter selection at the other end of the text.
- Click on the **Join** button of the menu window with the second mouse button. The source selection will now contain the parameter selection.
- Set the cursor at the position where the text should be inserted.
- Click on the **Copy** button or use the **F7** keyboard shortcut.

5.9 Copy, Swap and Move.

The **copy** and **swap** functions are used to copy, swap or move both text and expressions. The functions are available through the buttons **copy** and **swap** and the keyboard shortcuts **F7** and **F8**. To *copy* a part of text to a particular position, follow these steps:

- Select the text with the source selection.
- Set the cursor at the target position.
- Click on the **Copy** button or use the **F7** shortcut.

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

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

- Select one part with the target selection.
- Select the other part with the source selection.
- Click on the **Swap** button or use the **F8** shortcut.

If the target selection is empty, the swap function can be used as a *move* function. The cursor determines the target position of the move in that situation.

These three functions are also available to use with expressions if the selections have compatible types. The copy operation will not copy text to expressions and the swap and move operations will only work if both selections have the same type. The following table shows when the functions work.

Primary	Secondary	Copy?	Swap?
Text	Text	Yes	Yes
Text	Not a text	Yes	No
Expression	Expression	Yes	Yes
Expression	Operator, identifier	Yes	No
Expression	Text	No	No
Operator	Operator	Yes	Yes
Operator	Not an operator	No	No
Identifier	Identifier	Yes	Yes
Identifier	Not an identifier	No	No

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:

Primary	Secondary	Result
{\it italic \/}	italic	italic
$a \vee (b \wedge c)$	$b \wedge c$	$b \wedge c$
$R \vee S$	$P \Rightarrow R \vee S$	$P \Rightarrow P \Rightarrow R \vee S$

The difference between the target and source selection is either removed or copied, according to which selection is larger.

5.10 Undoing Changes

Mathpad has no functions to undo changes. For the moment, deleted parts of text and removed expressions are stored in the *buffer*. The contents of this buffer is displayed in the *buffer window*, as shown in figure 5.3. This window is used like a normal edit window and deleted text and expressions can be restored by copying them back to an edit window.

New deletions are added at the beginning of the buffer, which is used as a stack. Opening the buffer window will show you waht has been deleted. The following functions may add something to the buffer:

- Copy, if the destination is not empty.
- Delete or backspace, if the selection 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 will get back via the kill buffer. You can clear the buffer also by clicking on the **Kill** button.

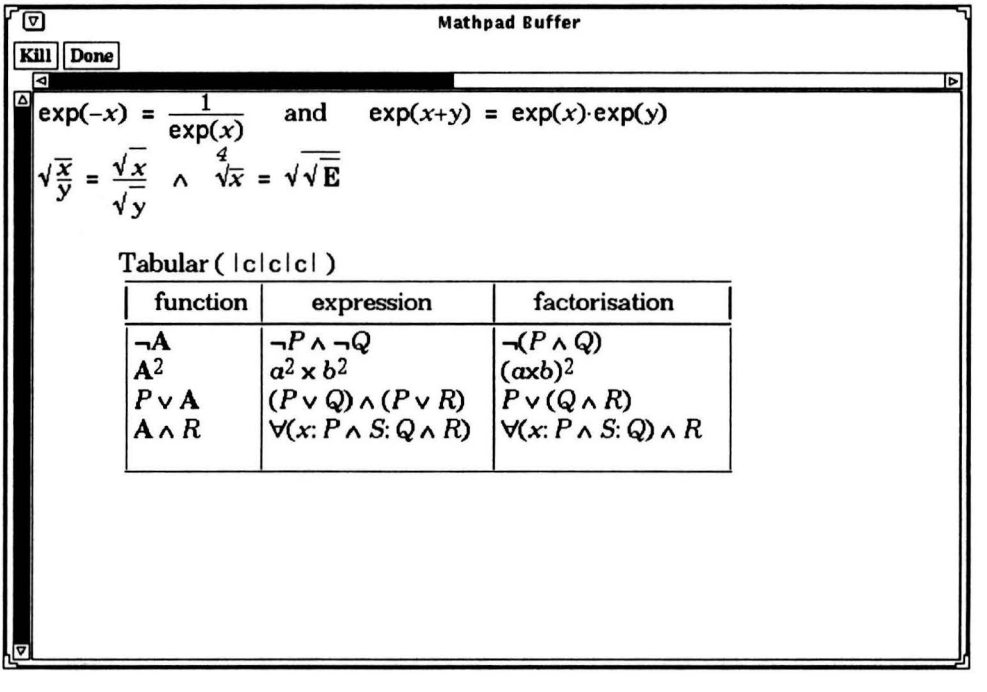


Figure 5.3: The buffer window

Chapter 6

Manipulating Expressions.

Mathpad is designed to be able to edit a mathematical document very fast. The manipulation of expressions is therefore very important. Most mathematical documents describe a certain topic and the same expressions are used repeatedly in a calculation or only small changes are made. The copy and swap functions, as described in section 5.9, are used for both text and expressions and are heavily used when (large) expressions are edited.

Often, mathematical rules are used to manipulate an expression. Mathpad has the following rules available:

- Reversal (Commutation).
- Identifier renaming.
- Applying functions.
- Distribution.
- Factorisation.

It is also possible to use the find-and-replace window to make a rule temporarily.

These rules and a number of other functions are described in the following subsections.

6.1 Operator Sequences and Parentheses.

In mathpad operator sequences are made by selecting the same stencil twice, so $a+b+c$ is entered by using two + operators. However, some sequences use different operators, like $0 \leq i < N$. To enter this kind of sequences the *ungroup* function is available through the **Ungroup** button. The function will insert the sequence of the selected subexpression as a subsequence in the larger expression. For example, the previous expression was made with the next steps:

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

The new structure of the ungrouped expression makes it possible to select more subsequences of the expression (for example $0 \leq i$).

The reversed action is also possible: a subsequence can be replaced with a subexpression that contains that sequence. This is done with the () or **Group** buttons. The () button will insert parentheses and places the sequence in a subexpression if needed. The **Group** function will only place the subsequence in a subexpression. For example, in the boolean expression $(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow a \Rightarrow c$, the last two implications are part of one sequences. In order to show the structure explicit, the part $a \Rightarrow c$ can be placed between parentheses with the () button, which results in $(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow (a \Rightarrow c)$. The other solution is using the **Group** button, giving $(a \Rightarrow b) \wedge (b \Rightarrow c) \Rightarrow a \Rightarrow c$, where the spacing shows the structure of the expression.

The () button can be used to put parentheses around the selected expression. If there are already parentheses, an attempt is made to remove them. This

will fail if the result would be easy to misunderstand, as in writing $a \times b+c$ when $a \times (b+c)$ is meant. You can insist on removal by pressing the control key at the same time or by using the **F4** key.

6.2 Changing the Layout.

Mathpad does no automatic layout formatting. Especially formatting large expressions is very difficult and every user has their own way of breaking these. So, instead of providing a difficult 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 made 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 follow the two steps:

- Select the *operator* or *template* with the target selection.
- Click on the **Version** button of the menu window.

A new version will now be used if possible. This version is the next version which contains the filled place holders.

It is possible to use an available version directly without moving through a row of versions. By clicking with the third mouse button on the **Version** button, a popup will appear with the available versions. The current version is marked with a star and one of the available versions can be selected. If a version is selected which does not contain a filled place holder, that argument will be removed.

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 set some text in another font. After selecting the template, the versions can be used to select another font (from **bold** to **typewriter**) or to change between $\sum_{i=0}^{N} i$ and $\sum_{i=0}^{N} i$.

Spacing around operators.

Mathpad uses the spacing around the operators to display the structure of the expressions. A good algorithm that works in all situations is not available and a facility is provided to change the spacing in situations where it is not satisfactory. The following keys can be used for this:

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

These changes are relative to the space mathpad calculated, which depends upon the structure of the expression and the use of parentheses. This fine tuning is usually not needed until you have finished writing your document and are preparing the final output.

6.3 Reverse

Often, the arguments of an operator sequence 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 stencil or operator sequence, selected by the *target selection*, at once. Especially with long sequences this function will be very welcome. The expression

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

can be transformed in

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

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 are still at the correct positions although they need to be changed sometimes. 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 selection expression has no subexpressions, the expression will be swapped with the expression left of the selected one (right if left is not available). By default, mathpad 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.

6.4 Renaming identifiers

Identifier renaming can be used to change all occurrences of an identifier in an expression at once. An instance of a general expression is made by copying the expression and renaming a particular identifier.

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.
- Select the scope of the rename function with the parameter selection.
- Click on the **Rename** button.

The identifiers within the scope are now renamed.

The renaming function will check if an identifier is bounded to an expression. The identifiers listed in the variable place holder are bounded and a rename will skip the expression if the old name is contained in this list.

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 bounded 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, $k^3{>}0$

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 i and k or by selecting the bound i, a 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 good document, this kind of expression should be avoided.

The last example looks like a normal rename but the function can be called in two ways. The first i is inserted with the **CM-V** command and is the only element of a variable list. The expression is therefore bound to this variable. Renaming this i will change all the i's used in the expressions in that text.

6.5 Selecting Functions.

In calculations functions are often used. In mathpad, 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 parameter selection.

The argument must be part of the body and the parameter selection is therefore part of the source selection. A body of the function can be any expression and the argument does not need to be empty. So, the following expressions are all functions, where the argument is marked with the **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-and-replace feature can be used.

6.6 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 parameter selection, the function is applied to the target selection by clicking on the **Apply** button. For example (the argument is given as $\mathbf{A}$):

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

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 expression.

6.7 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** button in the menu window. 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. For example:

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

The original function will be removed if you do not use the parameter selection, but let the target selection be both "victim" and argument. If you have

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

with the whole expression in the target selection, the $b{+}c$ part in the source selection and no parameter selection, then the result of the distribution will be

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

just as one should expect.

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

6.8 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** button of the menu window. To be able to factorise a function and operator sequence are needed. The function must be selected with the source and parameter selection while the operator sequence is selected with the target selection. After clicking on the button the factorisation will be performed. Some examples are:

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

When the function is not found in an expression from the template or operator sequence the expression will not be changed. If the function is not found at all, nothing will change. In the last example $\wedge R$ is not found in the first expression which will not change. In the second expression it is found so it will be removed and applied to the selected stencil. Distribution is therefore not the inverse of factorisation.

Chapter 7

Symbol and Stencil Palettes.

Mathpad uses a palette to give you the possibility to select symbols and stencils very easy. A *palette* is a window that contains a list of items and different palettes with the same contents can be opened. You can open as many palettes as you want in any size you want. The contents will be adjusted to fit in the palette.

There are two different palettes, the symbol palette shows a page of symbols and the stencil palette shows a set of templates. Both symbols and templates are selected in the same way: just click on them and you get a copy at the position of the cursor.

7.1 Special Symbols

To use the special symbols, the *symbol palette* should be open. This is done by clicking on the Symbol button in the main menu. It is possible to open multiple symbol palettes with different pages and different sizes. When a window is resized, the symbols will be placed such that they are readable and spaced as widely as possible. A symbol palette, as shown in figure 7.1, contains three buttons and at most 128 special symbols. At the top of the palette, the following buttons are available:

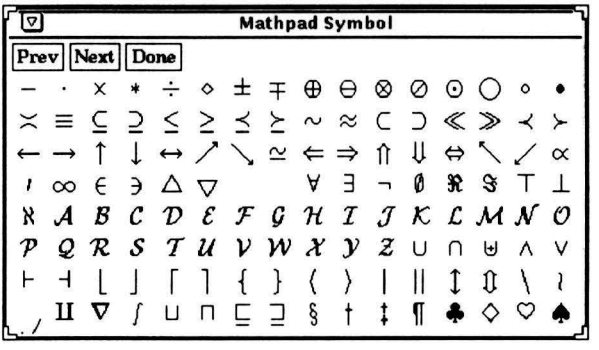


Figure 7.1: A symbol palette

Prev Go to the previous page with symbols.

Next Go to the next page with symbols.

Done Close the palette.

The buttons **Prev** and **Next** can be used to select different pages. This way multiple palettes with different pages can be opened.

Which pages are available depends on your local installation of mathpad. It is always possible to add more pages (up to a certain level). How you add fonts, pages or symbols is not yet described in this manual..

A symbol can be selected with the mouse by clicking on it. It's also possible to drag 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 filename.

When the LaTeX -file is created, every occurrence of a symbol will be replaced by the LaTeX -macro that is loaded when the editor is started. Mathpad will automatically select the correct LaTeX mode and mode braces (**$**) are usually not needed. These macros can be changed if necessary and you can add your own macro to a symbol with a special file (with extention .mpt)..

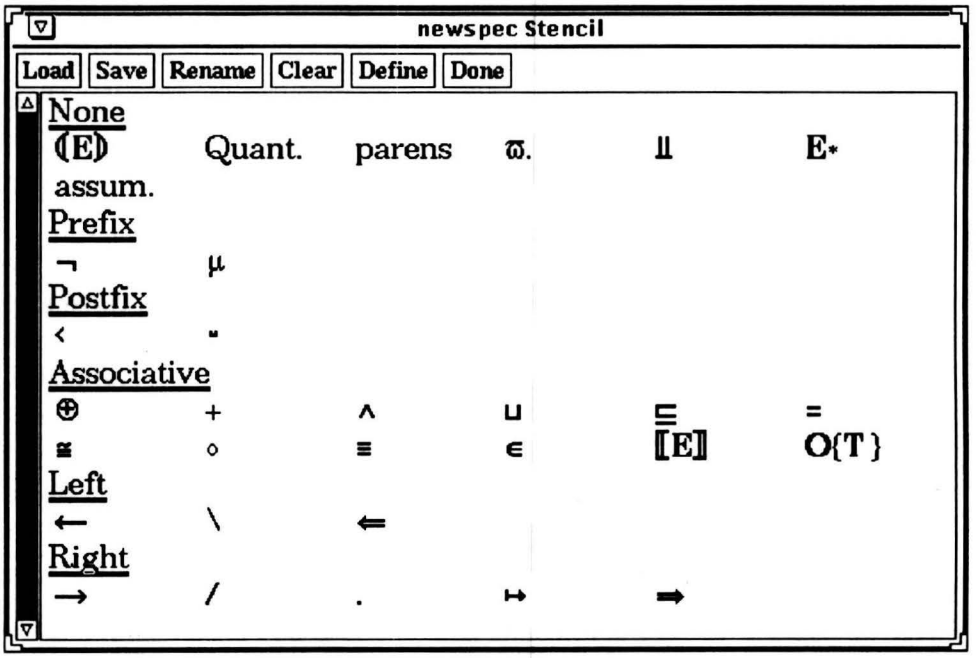


Figure 7.2: A stencil palette

7.2 Stencils

A stencil, as shown in figure 7.2, can be opened by clicking in the **Stencil** button of the main menu. If you have already a number of loaded stencils, a pop-up will appear with the loaded files and the options **New window** and **Cancel**. A star behind the file name indicates that the particular file is already visible in a stencil. When a file name is selected, the corresponding stencil will be opened. When **New window** is selected a new, empty stencil is opened.

It is possible to open multiple stencils with different sets of templates. This makes it possible to select different templates without using the scrollbar and to load different stencil files. The templates are displayed in columns and when the window is resized the number of columns will be adjusted to the new size. The scrollbar makes it possible to move the templates up or

down.

At the top of the palette six buttons are available to load, save, rename, clear or define templates or to close the stencil. The exact meaning of the buttons is

Load Remove the contents of the window and load the stencil from a specified file. The editor will first search for the file in the list of already loaded files. If the file is already loaded, the loaded file will be used unless the window already contains that file, in which case the file is reloaded from disk. If the file is not loaded, the editor will load it from disk. However, it is not possible to loaded two files from different directories with the same filename.

When the specified file is in fact a directory a popup will appear with the contents of that directory. This popup can be used like the popup that appears when a document is loaded.

Save Save the stencil in a specified file. This file will be placed in the stencil directory of the user.

Rename Rename the file that is loaded in that palette. This can be necessary if a file with the same name must be loaded from another directory.

Clear Remove the file from this palette. If this palette is the last palette that contains that file the particular file will be removed from memory. A pop-up will appear when this happens.

Define Open the define palette to create new templates or to change existing templates. The define window will be connected to this file and new templates will be added to this file. If the define window is already open, its contents will not be changed. The define window can be used to move templates from one file to another.

Done Close the stencil and the define window if necessary. The file and templates from the palette will not be removed from memory.

Templates are divided into separate groups according to their kind. Inside a

group the templates can be sorted by moving them left or right. The **M-left** and **M-right** keys can be used for this.

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 right mouse button must be used to select the template. The popup menu that appears contains the different versions and the correct version can be selected.

The missing place holders will be added when the template is an operator and the place holders are necessary. So, when you want to enter $a+b$, just click on the associative + operator and the two arguments are added. Then type '`a b` ' and the expression is finished. However, if you had selected an operator place holder, the arguments will not appear, only the operator is used.

Chapter 8

Find and Replace.

Every editor has a possibility to search for text and usually a replace function is also available. Mathpad is an editor and the incremental search and query replace are available for both text and stencils. These functions are based on the functions emacs provides and the editing facilities are limited. Therefor, a special window is available which makes it possible to search for complete expressions and to replace them with other expressions according to the entered rule.

8.1 The find & replace window.

The find and replace window, as shown in figure 8.1, looks like a normal edit window, except that it is divided into two equally sized subwindows. Each subwindow contains a separate document which is edited like a normal document. The upper document contains the expression or text to search for, the lower one contains the replacement.

At the top of the window, there are a number of buttons. These buttons are used to start searching or to replace an expression. The exact meaning of the buttons is:

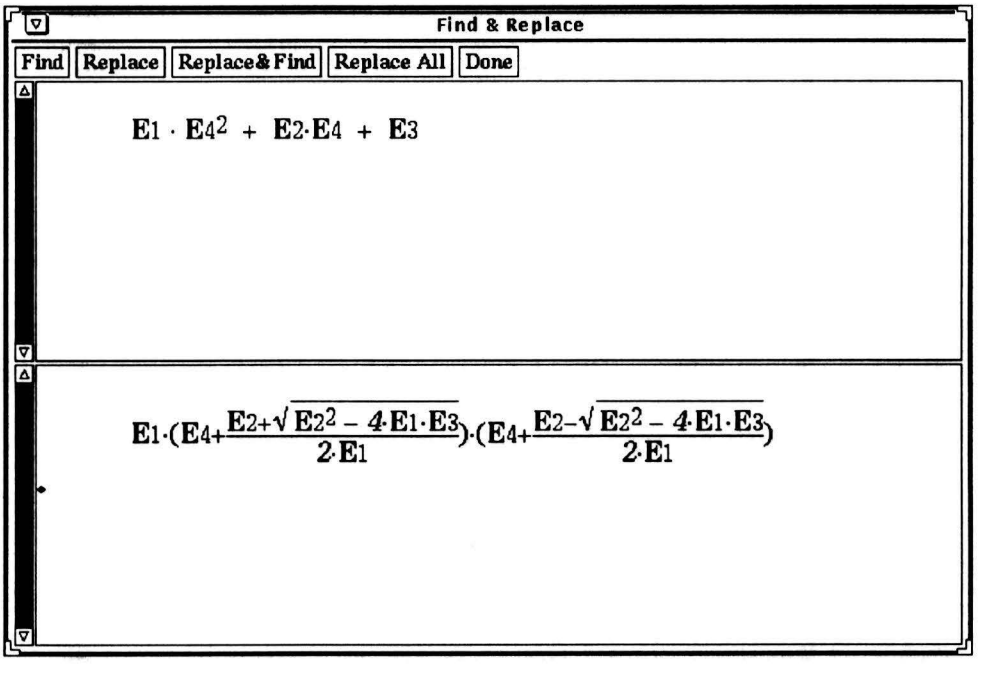


Figure 8.1: The find and replace window.

Find Search the expression in the upper document from the position of the target selection or cursor. Empty place holders with no indices have to match empty expressions and place holders with the same indices have to match the same expressions.

Replace Replace the found expression with the expression in the lower document. The indices of the place holders have to match in order to replace it correctly.

Replace&Find
Equal to performing the two functions **Replace** and **Find**.

Replace All Replace all occurrences of the upper expression with the lower expression.

Done Close the find and replace window and clear the two documents.

The find and replace functions are available for both text and expressions.

8.2 Special editing functions.

The place holders in the find and replace window have indices. These indices are used to make a connection between the different arguments and a place holder with no index will match only with an empty place holder. The indices are therefor very important.

Mathpad 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 indices from one place holder to another, swap two indices, reverse a sequence of place holders or distribute a place holder with index over any sequence.

If you to add a new index to a place holder, you can use the **C-I** key in combination with the universal argument. 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.

If you use the two constructions to find or replace something, the indices of the place holders are checked to ensure that the following weak restrictions hold.

- Two place holders with the same index have the same type.
- The place holders in the replace construction are also used in the find construction.

Both constructions can still contain multiple copies of the same place holder. So, 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 $\mathbf{E}_1 \wedge \mathbf{E}_2$.

Mathpad can only replace constructions if they have the same or a convertible type. Replacing text with an expression is possible, but 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 solved in the next release.

8.3 Some examples.

The find and replace window can be used for finding text, templates, identifiers, operators or complete expression structures. Together with the replace function spelling errors can be corrected and mathematic rules can be used. A number of examples show how it can be used.

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 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 part of text, just enter or copy the original text in the upper part and the replacement in the lower part, place the cursor

at the start position and click on the **Find** button. The first occurrence is now highlighted and the **Replace** button can be used to replace it. You can also use the **Replace All** button to replace all occurrences without asking. For example, if you want to replace 'some' with 'many', enter 'some' in the upper part and 'many' in the lower part of the window. The find function will search for 'some', which means that 'Some' is not found while 'some'one is. Replacing all occurrences is in this situation not smart. So, the **Find**, **Replace** and **Replace&Find** buttons should be used. In other situations, like replacing 'ccurence' with 'ccurrence', the **Replace All** button can be used without any problem.

The keyboard functions incremental search and query replace can also be used for this. However, these functions can't be interrupted as easy as the find and replace functions. While you are searching or replacing text, you often want to edit something. The functions from the find and replace window only use cursor position and the expressions in the two parts of the window. So, you can always edit the document between two find or replace actions. After you finished editing, just put the cursor at the start position and use the buttons from the find window.

If you change the screen format of a stencil, mathpad will not automatically replace old occurrences with the new version. The find and replace function can be used for this. Just copy the old version in the upper part and the new version in the lower part. Put the cursor at the begin of your document an a **Replace All** will do the rest. The replace can also be used to change from one stencil to a completely different stencil with different arguments.

Searching 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 keyboard command **CM-O** and then click 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 an y, type **CM-E x** in the empty upper part and **CM-E y** in the empty lower part. Put the cursor at the beginning of the document and click on the **Replace All** button. Now, every x is replaced with an y. The difference between this replace and the rename command is the fact that bounded x's are also renamed. Also the normal

query replace is different because that command will also change the normal x's from words like 'box' or 'expression'. Of course, if you type **x**, such an x is not renamed (note that **x** is not really a shortcut).

If you want to change large expressions according to a particular rule very often, you can decide to make the rule in the find and replace window 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 in mathpad. The upper part contains

$$\mathbf{E_1{\cdot}E_4}^2 \;+\; \mathbf{E_2{\cdot}E_4} \;+\; \mathbf{E_3}$$

while the lower part contains

$$\mathbf{E_1} \cdot (\mathbf{E_4} + \frac{\mathbf{E_2}+\sqrt{\mathbf{E_2}^2 - 4{\cdot}\mathbf{E_1{\cdot}E_3}}}{2{\cdot}\mathbf{E_1}}) \cdot (\mathbf{E_4} + \frac{\mathbf{E_2}-\sqrt{\mathbf{E_2}^2 - 4{\cdot}\mathbf{E_1{\cdot}E_3}}}{2{\cdot}\mathbf{E_1}})$$

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, mathpad 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 $-\ 5{\cdot}x$. Note that mathpad can match subexpressions like $x{+}y$ as arguments and that the parentheses and spacing from the find construction don't have to match. The parentheses and spacing in the replace construction will also be used

in the replacement. The find and replace can therefor be used to remove parentheses or add spacing.

In the future, mathpad should have an interface with a mathematic program like maple or mathematica. Then, the above manipulations can be done by that program.

Chapter 9

Defining Stencils

The editor has no built-in notational conventions and this can be a hurdle for a new user. There are however good reasons for not building in notational conventions.

First of all, the conventions that someone uses depend heavily on the research being undertaken by the user. Many groups have their own conventions and would prefer to see their own favourite ones built in. An editor written to cater for such preferences can only be used by a limited group in an easy way while other users are restricted by the defined conventions.

Secondly, the conventions used by an individual can also change during the course of time. A reason for a change can be the start of a new task or a technical development. For example, a summation over an interval can not be set with a normal type writer in a very nice way. One way to do this is

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

However with LaTeX it could be something like

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

and on a terminal with an extended ASCII font it could be

$$\sum(i \in [0,N] : i^2)$$

So, depending on the available resources, conventions can change.

Third, some operators have a different meaning and precedence depending on the context in which they are used. The mathematical + operator is sometimes also used as the boolean $\vee$ operator although the precedence is not equal. Were such an operator to be built-in, only one precedence could be used for that operator.

Fourth, the diversity of conventions and operators is so large that it is impossible to build all these conventions into the editor. When only a subset of these conventions is built in a facility is needed to create the missing ones. When this facility is easy to use, the user can also make the built-in conventions and change them if necessary.

Instead of the built-in conventions the editor has a powerful facility to create new "stencils" or to change existing ones, so every user can make his own set of notational conventions. Each stencil consists of a set of "templates" each of which may have several "versions". In the following subsections the process of creating and changing stencils is discussed.

9.1 The Define Window

A template can be defined in the *Define* window which is shown in figure 9.1. This window will be opened when the **Define** button in a stencil is used. The title of the window will indicate which stencil will be changed or expanded. The chosen stencil can be changed by clicking on the define button of any other stencil.

The buttons at the top of the window have the following function:

Install Install the new or modified template. Before installing the template

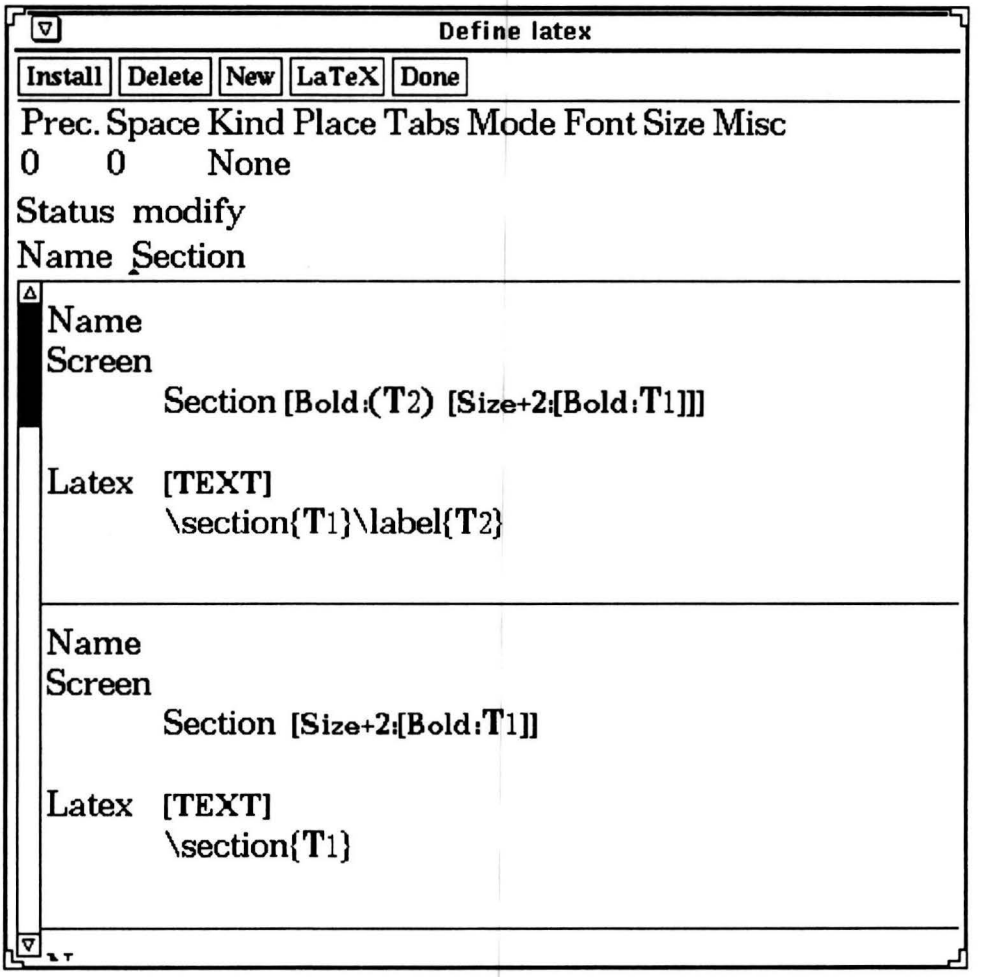


Figure 9.1: The define window

it will be checked on some points to ensure that the template will not cause any problems. These points are:

- There is at least one version.
- The template has a name, the first version has a name or the first version is visible in a stencil window. This check ensures that the template can be selected after it is installed.
- No screen format contains repeated occurrences of a place holder.
- The place holders in a LaTeX format all appear in the corresponding screen format. A LaTeX format can contain repeated occurrences of a place holder.
- There is one screen format which contains all the place holders from the template.

When a error occurs on one of these points, the cursor will be placed at the position where the error is found. Otherwise, depending on the status of the template, the existing template will be replaced by the modified one (**modify**) or the new template will be added after the last template of the same kind (**use** or **new**). After a successful installation the status will be set to **modify**.

Delete Remove a version or the template depending on the cursor. If the cursor is in one of the versions that version will be removed. Otherwise, the entire template will be deleted from the stencil, if possible. The contents of the define window will not be removed so the template can be reinstalled if it was deleted by mistake.

New Change the status and the contents. When the status is **modify** it will become **use** and the name of the template will be removed. When the status is **use** the status will become **new** and the template will be set to the empty template with no versions and no name. This way existing template can be used to create new ones.

Latex Make a LaTeX file that contains a backup of the defined template or complete stencil. This LaTeX file can be used as a hardcopy that describes how the different versions are made and what LaTeX they produce. When there is no version in the define window, the file will contain all the templates of the stencil that is selected. The name of the file can be entered in the pop-up that appears.

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

Below the buttons there is a list of items used to select special symbols or to enter some general information about the template. By clicking on these items, a popup menu will appear and a special symbol or value can be selected. These items are:

Prec. The precedence of the operator can be selected from the popup menu. The selected precedence will be displayed below the item. The precedence of a template with kind **None** is irrelevant and will be set to zero when the template is installed.

Space The space that will be placed around the operator can be set to a minimum default width. In the document this width can be changed by hand where needed. The selected width is displayed below the item.

Kind The type of the operator can be selected here. A template that will be used as an expression should have type **None** while a template that will be used as an operator must have another type according to the kind of operator (prefix, postfix, associative, left associative or right associative).

Place The different place holders.

Tabs The tabbing commands.

Mode The hints for the LaTeX mode.

Font The available fonts when mathpad was started.

Size The different sizes mathpad can handle.

Misc A number of special constructions, such as vertical stacking, tabbing environments and hidden parameters.

The meaning of the different special symbols is described in the next sections according to their group. The special symbols will be displayed in the screen

version in a small bold font, for example **[tab]** or **[Hide:]**. With the **M-T** key, these symbols can be switched off and the screen layout can be viewed.

Below these items the **status** is displayed, which can be changed by using the **New** button. The meaning of the different states:

modify The template in the define window already exists and will be changed. The template does not have to come from the stencil where the template will be inserted. Installing the template will replace the existing template if it was selected from the stencil where the defined template will be inserted.

use The template in the define window is a copy of an existing template. Often a template has the same precedence, spacing, kind and versions as another template except a specific symbol (for example: the sum and integral template of LaTeX). Installing the copy will create a new template.

new The template is not related to any other template.

The name of the template can be entered after the **Name** field. This name can only contain characters from the keyboard and it will be used in the stencil to display the template. If the name is empty, the name or the screen format of the first version will be used to display the template.

The different versions, displayed at the right of the scrollbar, can be edited like normal strings. For each version, the name of the version, the screen format and the LaTeX code are displayed. The name is used in the stencil and in the pop-up menu that shows the different versions. If the name is not entered, the stripped screen format will be used for this. The screen format is used in a document to display it. The LaTeX code determines how the template should be set in LaTeX , which can be made independent from the screen format.

9.2 Generation of LaTeX

When a template is used in a document, a copy of the screen format is inserted together with the precedence and kind. When LaTeX is generated, the copied information is used to find the correct LaTeX format. The LaTeX format of the best matching version will be used, where the best match is determined by:

- The information matches correct. That is, the screen format, the precedence and the kind match the copied information.
- The screen format and the kind match, but the precedence doesn't match.
- Only the screen format matches.

If more versions match to the same rule, the version of the last loaded or added template is used. If there are no matching versions at all, the automatic generated LaTeX will be used, which is usually only correct for short templates.

The screen format is very important when LaTeX is generated. If the screen format of a version is changed, the old instances of that version are not translated to LaTeX correctly. Therefore, the screen version of a template should not be changed too often and the old instances should be replaced with the new version. It is also possible to move the old versions to a separate stencil to be able to translate old documents or old instances to LaTeX .

This searching process is not very flexible. In a future release, the information will be stored in an other way to ensure correct LaTeX output.

9.3 Place holders

When a template is defined *place holders* are used to show the positions of the arguments in the template. Later these place holders can be filled according

to some rules. The five available types of place holders are selected from the popup menu which appears when the Place at the top of the window is used. The place holders are displayed on the screen with a bold character and the corresponding calligraphic character will be used in the LaTeX output. In the table below, each item contains the symbol displayed on the screen, the symbol used in the LaTeX output and the option available in the popup menu. The meaning of the different types of place holders are:

E, $\mathcal{E}$, 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, $\mathcal{O}$, operator
: An operator, replaceable by a template or an expression. An operator will be placed in the math mode of LaTeX unless the template specifies that it should be placed in the text mode. The spacing that TeX adds around operators will be removed by mathpad.

I, $\mathcal{I}$, 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. The identifier will be set in the math mode, where identifiers with only one character will use the math italic font from LaTeX while identifiers with more characters will use the normal italic font.

V, $\mathcal{V}$, variable
: A list of bound identifiers, separated by commas. The identifiers from this list can be renamed and the matching identifiers within the scope will be renamed also. The variable will also be set in the math mode.

T, , text
: A text which can in fact be a normal document. The text can contain any combination of text and expressions. Text will be set in the text mode and TeX will handle the line and page breaks. However, line breaking inside the tabbing environment must be done by the user.

Place holders can be inserted by selecting them from the popup menu or by using the keyboard shortcuts (**CM-E**, **CM-O**, **CM-I**, **CM-V**and **CM-T**).

The place holder will get a new index. It is possible to enter fifteen different place holders in one version.

These indices are used to make the connection between the screen formats and the LaTeX codes. The screen format can't contain any place holder twice. However, in the LaTeX code a place holder can be used several times which is useful at positions where output should be copied. To insert a copy of an existing place holder, the **C-U** command should be used with the correct index. For example,

C-U 3 E

will insert a place holder with index 3 in the LaTeX code. The command can also be use in the screen format, but the index 3 will only be used when it not already used.

An example.

Many documents start with almost the same preamble which contains only a few arguments such as the title and some style files. A simple template can insert the complete preamble where place holders are used for the arguments. For example:

Name: Preamble
Screen: Article
Styles: $\mathbf{T}_1$
Title: $\mathbf{T}_2$
Latex: `[TEXT]\documentstyle[mathpad,`$\mathbf{T}_1$`]{article}`
`\input{mpmacros}`
`\title{`$\mathbf{T}_2$`}\author{<Your name>}`
`\maketitle\begin{document}`

This template is not complete. A number of 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 in every document, are now inserted by using a template. The arguments are directly visible and the preamble of the mathpad document shows only what is necessary.

With the more powerful features, the title can be set in a large bold font and the used style files could be hidden in a special version. How these features are use is discussed in the following subsections.

9.4 Tabbing commands.

To make a good layout on the screen and also with LaTeX a tabbing environment is used. The **tabbing** environment used by LaTeX , as described in appendix C.9.1 (pages 179–181) of the LaTeX user's guide & reference manual, has been adapted to make it more robust. This resulted in another tabbing environment called **mpdisplay**. The commands that are used are almost the same as used by the normal **tabbing** environment. The **mpdisplay** environment, however, has four arguments with the following meaning:

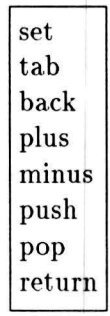
1. The unit size of the space placed around operators.

2. The size of the space used when no tab is defined.

3. The size of the space between two lines.

4. The number of tabs the left margin must be moved to the right.

The editor will insert these arguments when the document is converted to LaTeX and they can be changed via the default window. These arguments are necessary to be able to make local adjustments after the LaTeX is produced.



The tabbing commands can be selected from the popup menu that appears when the **Tabs** item at the top of the window is used. The meaning of these tab commands is:

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 a tab stop will be defined. The length of the space is given by argument 2 of the **mpdisplay** environment. The tab command from the normal tabbing environment will report an error the tab stop is not defined.

back Move to the previous tab stop (\<). This command can only by used at the start 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 **\pushtab** command from LaTeX which doesn't work well 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 LaTeX output. When a line in the LaTeX code of a version gets very long and it will be typeset in the tabbing environment, this return can be used to break the line. The normal newline will insert a `\\` when it is used within the tabbing environment.

There are shortcuts for the commands **set** (**C-tab**), **tab** (**tab**), **back** (**S-tab**) and **return** (**C-return**).

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. The label will be numbered with an equation number and the number will be placed at the left side between parentheses. Each line can be labelled once.

The tabbing commands should be used carefully because templates will get nested when they are used. The following rules can be used to ensure that nothing will go wrong with the tabbing commands. It is always possible to ignore the rules at your own risk.

- Use the **push** and **pop** commands in matching pairs. Every **push** should be followed by a **pop** within the same template.
- Use the same number of **plus** and **minus** commands within a version. The order is not always important because the mpdisplay environment will start a number of predefined tab positions.
- Move the margin forward to a (text) place holder if the version uses multiple lines. The place holder could be replaced by something which uses multiple lines and could mess up the layout.
- Try to avoid using the **back** command. Usually the same effect can be achieved by using a **minus** command in the previous line and a **plus** command in the same line. The **back** command could result in strange cursor reactions or strange selections.
- To be sure that the used tab stops in a template will not change the tab stops from other templates, a **push** and **pop** command should be added at the begin and the end of each version with tabbing commands.

Some examples.

Defining a template which uses only one line is not so difficult: just place the different symbols behind each other. It gets more difficult if a template uses several lines. The program construction **do S od** can be used on one line if the statement **S** is not to large. However, if the statement is large, there should be a way to show it correct, t.i. the **od** positioned exactly below the **do** and the statement behind the **do**. This can be done with the next template.

Name: do od (short)
Screen: do $\mathbf{E}_1$ od
Latex: `{\sf do}~E1~{\sf od}`

Name: do od (long)
Screen: [push]do [set][plus]$\mathbf{E}_1$[minus]
od[pop]
Latex:
```
\push${\sf do}~~$\=\+$E1$\-\\
${\sf od}$\pop
```

The first version is quiet simple, although the screen format could contain codes to set do and od in a sans serif font. 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 that where already defined. The tab position is defined to be able to move the margin just before the expression place holder. The margin is moved forward with the [plus] and is moved back with the [minus], which is done to ensure that the layout will be correct even if the expression uses several lines.

The above template is used in combination with a guard ($\mathbf{E}_1 \rightarrow \mathbf{E}_2$) and a guard separator ($\Box$). The separator is usually placed in the following way:

$$\textsf{do}\ \ a \wedge b \ \rightarrow\ \textsf{do}\ \neg b \ \rightarrow\ a,b := b,a \ \Box\ a \ \rightarrow\ b := \neg b\ \textsf{od}$$
$$\Box\ \ a \vee b \ \rightarrow\ b := a$$

od

The separator can be used when the next operator is available:

Name: Separator (short)
Screen: □
Latex: `$\Box$`

Name: Separator (long)
Screen: [push][minus]
□[tab][plus][pop]
Latex: `\push\-\\`
`$\Box$\>\+\pop`

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 S od** template, which moves the margin forward. The operator will move the margin backward first. Then the box can be placed exactly below the **do**. The [tab] will move to the old margin and the margin is move forward the ensure that the second argument has the same margin as the first one. Of course, this operator should not be used when the margin can't move backward.

The guard template should also have different versions, depending on the length of the two expressions. Two possible version could be:

Name: Guard (short)
Screen: [push]$\mathbf{E}_1$ → [set][plus] [set][plus]$\mathbf{E}_2$[minus][minus][pop]
Latex: `\push$E_1$~~\rightarrow~$\=\+$~$\=\+$E_2$\-\-\pop`

Name: Guard (long)
Screen: [push]$\mathbf{E}_1$ →

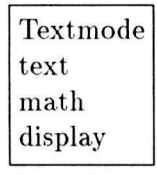
```
                [set][plus]  [set][plus]E2[minus][minus][pop]
Latex:  \push$E1~~\rightarrow$\\
        $~~~~$\=\+$~$\=\+$E2$\-\-\pop
```

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 statement separator will look like the separator template discussed earlier.

The first version will move the margin after the arrow. This is only possible if one of the two expressions is not very large. The second version will force the second expression to start at a new line and will indent a little to increase the readability. An other possible version could place the arrow on the second line too.

9.5 LaTeX Modes

LaTeX uses the 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. Normally, the **$** sign can be used to switch between text mode and math mode but some other commands can also be used. The **\mbox** command is used to change to LR mode, regardless of the mode it is used in. The **$$** command however will change to the displayed math mode if it is used in the text mode, but will have no effect in math mode and will change to the text mode if it is used in the displayed math mode. Mathpad will place the different symbols in the mode they can be used in. 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. It is possible to define templates that change the mode themself which would result in incorrect LaTeX code. Therefore, hints can be inserted in a template, so mathpad can keep track of the mode at any moment.



The popup menu above is used to insert the different hints. The popup will appear when the **Mode** item is used on the top of the window. The meaning of the hints is:

Textmode
: The template must be set in the text mode. Normally, a template will be set in math mode. This hint can only be used at the first position of a version.

text
: The LaTeX code has changed to the text mode. After this hint the code will be set in the text mode until another hint is used.

math
: The code has entered the math mode and the rest of the code should be placed in the math mode as much as possible.

display
: The code has entered the displayed math mode. The text used in this mode will be put in a `\mbox{}` command. The rest of the code will be placed in the math mode.

Except for the **Textmode** hint, the hints can't be used at the first position of a template. A hint should not be used to remove a `$` sign that mathpad added automatically because the `$` sign will not always appear in the LaTeX output. The `$` signs around a place holder will be inserted only if the contents of the place holder must be placed in another mode which depends on the used symbols or templates.

9.6 Fonts.

Stencils can use different fonts to add a WYSIWYG look to it. Many constructions from LaTeX use different fonts or typestyles of some parts for the

construction. The Theorem environment has a bold (and large) header, the items in the description are bold and the verbatim environment uses the typewriter font. Also normal font switches are often used to highlight words, usually italic, slanted, bold or sans serif. In mathpad, all these constructions can be inserted and the font switches can be added.

Which fonts are available depends on the local installation but every user can add his own fonts. The file **fonts.mpt** is loaded when mathpad is started. The file describes which fonts are available and how they should be converted to LaTeX . These fonts will be listed in the popup menu that appears when the **Font** item on the top of the window is used. With a small number of entrys the popup could look like:

A font can be selected by clicking on it. Then, two special symbols are entered at the position of the cursor: one to open the specific font and one to close it. For the italic font, these symbols will look like **[Italic:** and] in the screen format and `{\it` and `\/}` in the LaTeX code, which is specified in the **fonts.mpt** file. These two symbols are connected and will be removed together to ensure a matching pair within a version.

An example.

The following example uses the type writer font to simulate the verbatim environment of LaTeX . In this environment the output of LaTeX will be exactly the text that is entered, which is normally used for typed text.

Name: Short verbatim+
Screen: [TypeWriter:T_1][Hide:+]
Latex: [TEXT]\verb+T_1+

Name: Long verbatim
Screen: [TypeWriter:T_1][Hide:verbatim]
Latex: [TEXT]\begin{verbatim}T_1\end{verbatim}

The first version can be used to make a short text verbatim, for example a LaTeX command. The text can not contain a plus because the plus will end the verbatim in LaTeX . This can be solved by adding a version where the plus is replaced by a minus.

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.

Mathpad will not check which environment or font is used. If the text in one of these versions contains a special symbol like π, LaTeX will produce **π**, not π. (In mathpad, you could see: ...produce π, not π.) This can be used to show what LaTeX code is used for a construction without actually typing it.

The **[Hide:]** command is used to make a difference between the two versions. Mathpad uses the screen version to make a link with the LaTeX code that must be used. This command will be explained later.

9.7 Sizes.

Also switching between different sizes can be used to make a template more WYSIWYG, but it can also be used to show the difference between versions or between templates. The different sizes can be selected from the popup

menu that appears when the Size item is clicked on. The popup contains a range of numbers which indicate how large the new font should be with respect to the current one. Selecting a negative number will decrease the size of the font by a number of steps. A positive number will increase the size. It is possible to nest these size changes which is useful when the size of a font should depend on the level of the template (for example a subscript).

After a size change, the used font will be the default text font. This behaviour is based on the way LaTeX combines size changes and font changes. So, if a small bold font is needed, you have to change the size to small first and change the font to bold later.

The special symbols to change the size are displayed in the screen format with [**Size+*n*:** and] which determine the scope of the size. The number n will be the number that is selected from the popup menu. The LaTeX code will contain the macros {`\size` and } where `\size` is one of the size-changing commands from LaTeX .

An example.

A section header is type set in a larger font. In mathpad, this can be done with the following template.

Name: Section

Screen:

Section [**Size+2:[Bold:**$\mathbf{T_1}$]]

Latex: [**TEXT**]

`\section{`$\mathbf{T_1}$`}`

This template used 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.

9.8 Special constructions

With the previous commands, many constructions can be made. There are however a number of special mathematic construction which can't be made yet. For example, superscripts, fractions and quantifiers have symbols at unusual positions. To be able to display these constructions on the screen in a satisfying way, the stacking construction can be used.

When a number of items is listed, the margin is moved to the right and a symbol is used to indicate the different items, for example:

- This is the first item of this example. The margin is positioned at the right of the dot, just before this text.

- However, LaTeX uses no margins in this situation. Instead, special environments and the `\item` macro are used to achieve this.

In LaTeX , there are several environments that use the margin and it is possible to make more environments which do the same. Instead of building in the standard LaTeX environments, the general tabbing environment can be used on the screen while the correct LaTeX output can be used in the LaTeX code.

In some situations the LaTeX output contains text that is not used at the position where it occurs, for example the text from a footnote or an index. The text should not be displayed on the screen if it is not wanted. The possibility of adding hidden parameters to a template can be used for this.

The special symbols used for these constructions are available through the following popup menu which appears when the Misc item is selected at the top of the define window.

Stack Base
Stack Center
Tabbing
Display
Hide
Line
Space
Dots
Bar

The meaning of these symbols will be discussed in the following subsections.

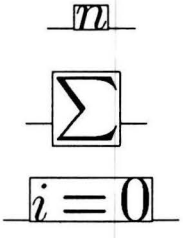
9.8.1 Stacking expressions

In LaTeX there are different ways to put things above each other. To be able to do the same thing in mathpad, two general constructions are used to simulate this. They both contain three arguments which will be placed above each other.

Every argument will define a box with a *width*, *height*, *baseline*, *ascender* and *descender*. The left side of the expression

$$\sum_{i=0}^{n} i$$

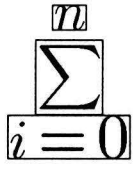
is built with the following boxes



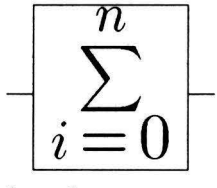
The box with the little line at the left side indicate the size and position of the contents of the box in the following way:

- the *width* and *height* of the box are the width and height of the text inside the box.
- the *baseline* of the box is the little line at the left side.
- the *ascender* of the box is the part of the box above the baseline. The vertical size of this part is called the *ascent*. The ascent can be negative if the box is entirely below the baseline.
- the *descender* of the box is the part of the box below the baseline. The vertical size of the descender is called the *descent*, which is negative if the box is entirely above the baseline.

With the stack construction of mathpad these three boxes can be put on top of each other, like this:



The boxes will be centered unless special codes are used to move a box to the left or to the right side. 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, which is usually done with an optional argument such as `[t]`, `[b]` or `[c]`. In mathpad, the position of the baseline can also be changed. However, the baseline can only be set to two positions:

- at the same position as the baseline of the middle box.
- a half **ex** below the baseline of the middle box (**ex** is the height of an x).

This extra option makes it possible to create subscripts, superscripts, fractions, small vectors and possibly many other constructions.

The meaning of the special codes from the popup menu which are used for the stacking of different symbols is:

Stack Base Insert a stack with three fields. The baseline of the stacked symbols will be the baseline of the middle field. On screen, the stack will look like **[StackB: : :]** where the semicolons separate the three fields. The 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 not be generated by mathpad.

Stack Center This stack will be almost the same as the stack described above. The only difference is the position of the baseline, which will be a half **ex** below the baseline of the middle field. On screen this stack will look like **[StackC: : :]**.

Line Insert a variable 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 line will be display with **[Line]** on screen while the LaTeX will be `\hrulefill`.

Space Insert a variable space. Just like the line, the length will depend on the width of the other fields. The variable space can be used to align the different fields left or right by inserting the space at the correct position. The space is displayed with **[Sp]** and the LaTeX will be `\hfill`.

Dots Insert a variable dashed line. The length of this horizontal line will depend on the length of the other fields. This line is displayed with **[Dots]**, the LaTeX will be `\dotfill`.

Bar Insert a vertical line. The length of the line will be the height of the field it is used in. The bar can also be used outside a stack, in which case the height of the line will be used. The bar is displayed with **[Bar]**, the LaTeX will be `|`.

The length of a variable line or space will be the difference between the longest field and the field the line or space is used in. If multiple lines or spaces are used inside one field, the available length will be divided across the different lines and spaces.

Mathpad centers the three fields of a stack, which is equal to adding a space at the left and right of every field. If a field should 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 centered 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

With the stack, many difficult constructions of LaTeX can be simulated. To show how this is done, some examples are described here. Each example starts with a definition followed by an explanation on how to use it and what possible changes are possible.

Name: Subscript
Screen: **E_1[Size+-1:[StackC: ::E_2]]**
Latex: **$$E_1$_{$E_2$}$$**

This template can be used in the math mode and the **[Size+-1:]** code is used to make the subscript one size smaller. This way, it is possible to add subscript to subscript.

If the first expression is removed, the subscript can be used as an postfix operator. Depending on how the template is used, **{}**should be added before the **_** in the LaTeX code.

The space in the first field is used to be able to select the template with a single click. Clicking above the subscript will now 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.

Name: Displayed sum
Screen: [StackB:[Size+-1:E_2]:[Size+4:$\sum$]:[Size+-1:E_1]] E_3
Latex: `$\displaystyle{\sum_{E1}^{E2}}\ E3$`

Name: Inline sum
Screen: $\sum$[StackC:[Size+-1:E_2[Sp]]: :[Size+-1:E_1[Sp]]] E_3
Latex: `$\textstyle{\sum_{E1}^{E2}}\ E3$`

These two versions of the sum template can be used to make $\sum_{\mathcal{E}}^{\mathcal{E}} \mathcal{E}$ and $\sum_{\mathcal{E}}^{\mathcal{E}} \mathcal{E}$.
The first version uses a very large summation symbol and places two smaller arguments above and below is. The three fields of the stack are centered. The LaTeX code contains the `\displaystyle` macro which is used to force LaTeX to place the two argument above and below the sign instead of to the right. With mathpad it is possible to use a displayed summation as one of the argument of another summation. The size of this summation will be smaller in mathpad 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 [Sp] 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. Mathpad 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.

Name: Underline text
Screen: [StackB::T_1:[Line]]

Latex: [TEXT]\underline{T_1}

Name: Underline expression
Screen: [StackB::E_2:[Line]]
Latex: $\underline{$E_2$}$

Name: Overline
Screen: [StackB:[Line]:E_2:]
Latex: $\overline{$E_2$}$

Name: Underline both
Screen: [StackB::T_1 E_2:[Line]]
Latex: [TEXT]\underline{$T_1$$E_2$$}

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 to long. Mathpad 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 nice but it is more difficult to make it with LaTeX . With mathpad, just select the parts and use the underline template.

Name: Box
Screen: [StackB:[Line]:[Bar] T_1 [Bar]:[Line]]
Latex: [TEXT]\fbox{T_1}

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 to large to ensure a correct LaTeX output and a correct layout by mathpad.

The template could be better by adding the minipage and tabbing environments of LaTeX . The layout of LaTeX will then be correct in almost every case. Mathpad will still have some trouble displaying it correct.

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.

Name: Displayed fraction
Screen: [StackC:E_1:[Line]:E_2]
Latex: `$\displaystyle{\frac{E1}{E2}}$`

Name: Inline fraction
Screen: [Size+-2:[StackC:E_1:[Line]:E_2]]
Latex: `$\textstyle{\frac{E1}{E2}}$`

Also the fraction can be made for both the displayed and the inline version. The fraction can be used nested like almost all template. 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 mathpad. Both inline and displayed fractions are used and changing between them is very simple. Of course, LaTeX has to support

the construction you made but it will never be much different from the expression the mathpad displays.

In some situation the produced LaTeX code is not optimal. The style commands are not always necessary but they make sure the layout of LaTeX will resemble the layout of mathpad.

Name: Root
Screen: [StackB: [Line]:$\sqrt{}$ E_1]
Latex: $\sqrt{E_1}$

Name: n-th root
Screen: [StackB:[Size+-2:E_2]:[Sp]$\sqrt{}$:][StackB:[Line]:E_1:]
Latex: $\sqrt[E_2]{E_1}$

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 mend with the expression

$$\sqrt{\frac{x}{y}} = \frac{\sqrt{x}}{\sqrt{y}} \quad \wedge \quad \sqrt[4]{x} = \sqrt{\sqrt{x}}$$

although is does not look completely WYSIWYG in mathpad. After all, mathpad is designed for editing text and flexible mathematics in an easy way.

Name: Vector
Screen: [StackB:$\rightarrow$:E_1:]
Latex: $\vec{E_1}$

Name: Long vector
Screen: [StackB:[line]$\rightarrow$:E_1:]
Latex: $\overrightarrow{E_1}$

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 centered with respect to the height. The line of the arrow is positioned in the middle, so the line will enlarge the arrow.

9.8.2 Using the tabbing environment

The tabbing environment is used in a **Display** expression, available through the main menu. Inside this tabbing environment the margin can be moved to the right or left which is very useful in lots of situations. Often the tabbing environment itself is not wanted in the LaTeX output but it could be useful on screen. The two special symbols **Tabbing** and **Display** from the popup can be used to enter and leave the tabbing environment from within a template. However, the LaTeX output does not have to enter and leave the tabbing environment. The full power of the tabbing commands can be used in the screen format of the template while the LaTeX code contains the correct code.

The meaning of the two special symbols is:

Tabbing Insert two symbols to open and close the **tabbing** environment. Between these to symbols the tabbing commands can be used, so tab stops can be set and the margin can be moved. The symbols are displayed with **[Tabbing:** and], the LaTeX code will be `\begin{tabbing}` and `\end{tabbing}`. The layout of LaTeX can differ from the layout on screen because mathpad uses the **mpdisplay** environment on screen.

Display Insert two symbols to open and close the **mpdisplay** environment. The **mpdisplay** environment is almost the same as the **tabbing** environment except a few commands and the position of the margin. The used symbols are **[Display:** and] on screen and the LaTeX code will be `\begin{mpdisplay}` and `\end{mpdisplay}`. The needed arguments will be added when the actual LaTeX file is made.

The two tabbing environment can be used nested. On screen and in LaTeX , a nested environment is equal to a pair of **push** and **pop**.

An example

The **itemize** environment moves the margin to the right and places dots (or other symbols) at the correct positions before the margin. Moving the margin and placing the dot is done by the `\item` command from LaTeX . In mathpad this environment can be made with the templates

Name: Itemize
Screen: [Tabbing:Item[set]ize
T_1]
Latex: [TEXT]`\begin{itemize}`
T_1
`\end{itemize}`

and

Name: Item
Screen: • [tab][plus]T_1[minus]

Latex: [TEXT]`\item` T_1

Of course, these templates should only be used in the correct combinations. 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 every thing, including the first template. It is possible to nest any number of these itemize environments within mathpad, although LaTeX could have trouble with it.

These two templates can also have different versions to be able to switch easy between the itemize, enumerate and description environments. One version of the item template shall have an extra argument to be able to enter the argument of the description item. So, with two templates the itemize, enumerate and description environments of LaTeX can be simulated in mathpad.

There are more environments which use the margin. They can be simulated in the same way with the **tabbing** or **mpdisplay** environments.

9.8.3 Hidden parameters

The last special symbol combination can be used for several things. The **[Hide:** and] symbols make it possible to at something to a template what will not be displayed on screen. A number of possible situation where this is very useful:

- Stencils can be very complex and a little comment can be added. This comment can be used as a hint for the one who made it or as an explanation for other users. Comments should not be to large.
- There may be versions with the same screen format. At the moment the LaTeX code of only one of these versions will be used. The **hide** construction can be used to make different screen formats with the same layout on screen. Therefore the correct LaTeX code will be used.
- Some LaTeX construction have arguments which will not produce any output at the position where the appear in the text. These arguments should not be visible in mathpad either, but you must be able to edit the argument. This problem can be solved by defining two versions of that template. One version displays the argument, so it can be edited. The other version hides the argument with the **hide** construction. Switching between the two versions will show or hide the specific argument.

At the moment, a newline inside a **hide** construction will still be visible, which results a number of empty lines.

An example.

The next template can be used to add an item to the index. The item can be the word that is selected or some related text.

Name: Index
Screen: [**Bold Italic:**T_1*]
Latex: [**TEXT**]T_1\index{T_1}

Name: Index Hide
Screen: [**Bold Italic:**T_1**][**Hide:**T_2]
Latex: [**TEXT**]T_1\index{T_2}

Name: Index Edit.
Screen: [**Bold Italic:**T_1*T_2*]
Latex: [**TEXT**]T_1\index{T_2}

The first version will just add the text to the index. The text will also be inserted in the normal text. This is one of the situations where the text needs to be inserted twice in the LaTeX output which is done by using the same place holder twice in the LaTeX code (using the **C-U 1 CM-T** key sequence).

The second version hides the text that will be placed in the index. LaTeX used the text only in the index and not at the position where the text is entered. The hide command will simulate that.

The third version makes it possible to edit the text that used be used in the index. Switching between the different versions makes it possible to show the text on demand.

All the versions use the bold italic font and one or two stars to highlight the text. This can of course be changed so make it look more WYSIWYG but

it should be possible to see where the index items are added. The first text place holder can be removed from the second and third version. The index does not depend on the text which is used to add the index item to.

9.9 Extra edit commands

There are some extra edit commands in the define window. These following commands are shortcuts for the special symbols from the popups:

Key	Symbol	Key	Symbol
Tab	[tab]	**C-?**	[Hide:]
S-Tab	[back]	**M-=**	[Stack?:::]
C-Tab	[set]	**CM-E**	$\mathbf{E}_n$
C-Return	[NwLn]	**CM-O**	$\mathbf{O}_n$
M- –	[Line]	**CM-I**	$\mathbf{I}_n$
M-space	[Sp]	**CM-V**	$\mathbf{V}_n$
M-.	[Dots]	**CM-T**	$\mathbf{T}_n$

The new version is always added at the end of the list and the first version will be used as an example for this new version. To be able to use an other version as an example or to add a new version between two existing versions, the following keys can be used to move the version with the cursor up or down:

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

These keys could not be available due to a bad keyboard installation or a virtual window manager. If this happens the keyboard can be redefined using mathpad own keyboard definition files.

The special codes which are inserted the popup menus add layout changes to the screen versions. It is possible to edit the screen version in two modes. The first mode displays the codes and does not interpreted them. The second mode interprets the codes and they will not be visible. Obvious, the first mode is used to edit the screen version while the second mode is used to preview the version. With the **M-T** key, you can switch between these two modes.

The next table contains the codes which use multiple symbols to define their scope or to seperate their fields.

Code	Symbols
Font change	[*name*: ,] where *name* is the name of a loaded font.
Size change	[**Size***n*: ,]
Stack Base	[**StackB:** , : , : ,]
Stack Center	[**StackC:** , : , : ,]
Tabbing	[**Tabbing:** ,]
Display	[**Display:** ,]
Hide	[**Hide:** ,]

These symbols are used as special braces and separators. The close braces look all the same but are in fact different. Mathpad will ensure that the different braces are used in matching pairs and are nested correctly. To do this, the remove and transpose functions work a little different.

If you remove one of these symbols, the other symbols for the specific code are also removed. For example, if you have the construction '**[Bold:**Section]' and you remove ']', the result will be 'Section'. Or, if you have '**[StackC:[Line]:[Bar]**-Box**[Bar]:[Line]]**' and you remove a ':', the result is '**[Line][Bar]**Box**[Bar][Line]**'.

The transpose function (**C-T**) works a little different to ensure that the symbols are nested correctly. First, the symbols at the left and right side of the cursor are moved according to the following rules:

- If one of the two symbols around the cursor is not part of a special code, t.i. the symbol is not listed in the above table, the transpose will

swap the two symbols and move the cursor one position forward (just like the normal transpose).

- If the right symbol is an opening brace, the left symbol and cursor are moved behind the matching close brace.
- If the left symbol is an close brace, the right symbol will be moved before the matching opening brace.
- Otherwise, no symbols are moved and the cursor is moved one position forward.

A few examples will show how the transpose can be used, where the cursor is marked with $_\diamond$.

Before	After transposing	Rule
[Bold:]$_\diamond$word	**[Bold:**w**]**$_\diamond$ord	1
[Size+1:]$_\diamond$**[Bold:**Section**]**	**[Size+1:[Bold:**Section**]]**$_\diamond$	2
[StackB:-::$_\diamond$**[Hide:**$\mathbf{T_1}$**]]**	**[StackB:-:[Hide:**$\mathbf{T_1}$**]:**$_\diamond$**]**	2
[Size+-1:$_\diamond$**[StackC:::**$\mathbf{E_1}$**]]**	**[StackC:::**$\mathbf{E_1}$**][Size+-1:**$_\diamond$**]**	2
[Size+1:[Bold: Section**]**$_\diamond$**]**	**[Size+1:][Bold:**Section**]**$_\diamond$	3
[Italic:$_\diamond$**]**It	**[Italic:]**$_\diamond$It	4

The transpose is used when a part of a version must be set in another font or size.

Chapter 10

Changing defaults

The default window, as shown in figure 10.1, can be opened by clicking in the default button of the main menu. In this window a number of variables are displayed which can be changed. With these variables the user can change the layout on the screen, the layout of the LaTeX -code, the paths for the different files and the fonts that are used.

At the top of the window three buttons are available with the following meanings:

Set	Use the new values and change the layout if necessary. Some numerical values are checked to make sure that the values are reasonable.
Save	Use the new values and save these values in a file named '.mpdefaults'. This file will be loaded when the editor is started so the correct settings can be used.
Done	Close the default window and ignore the changes since the last use of the set or save button.

Below the buttons there are some fields which can be changed. At the left side of the window the descriptions of the fields are displayed, at the right side the values of these fields are displayed. The different fields are

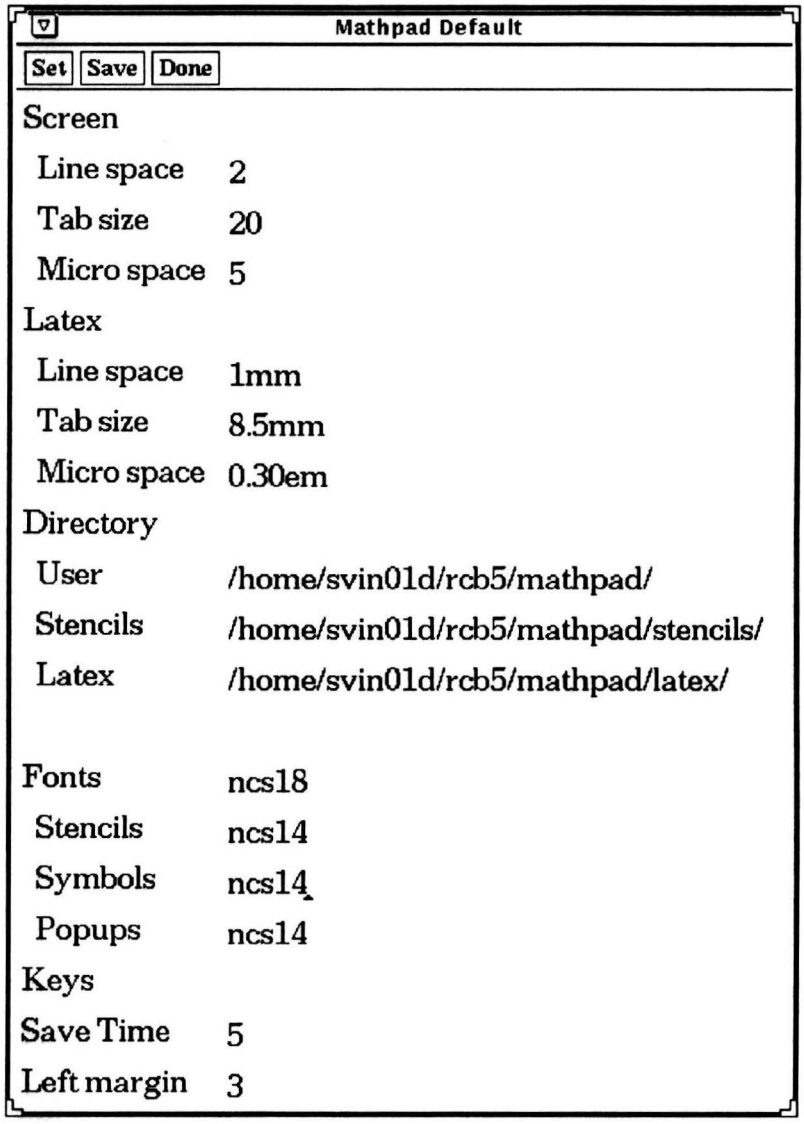


Figure 10.1: The default window

Screen With these 3 fields the layout on the screen is defined. These values can be adjusted so that the layout corresponds with the layout that LaTeX produces. The values are numerical and the unit is screen pixels.

Line space
The distance between two lines.

Tab size The default distance between two tab-positions.

Micro space
The length of one micro space, used around operators.

Latex These 3 fields define the layout that LaTeX will produce. Every field contains a string which will be placed in the LaTeX -code and it should define a length, for example 12.7mm or 0.5in. These strings will not be checked so the user should be careful. The meaning of the fields is the same as the meaning for the screen fields.

Directory There are 3 directories the user can change. These directories are used to store the different files or to search for files. The name of a file will be put behind the directory name so a '/' at the end is usually desired. The different files will be allocated to the directories in the following way:

User The documents.

Stencils The stencils.

Latex The LaTeX -files .

Fonts This name is used to select a group of fonts. When the **Mathpad** is started the file with that name will be used to load the different fonts. The specified group of font will be used in the edit windows, the buffer window, the find & replace window and the lower part of the define window. The fonts for the stencil window, the symbol window and the default window can be specified separately:

Stencil The fonts for the stencil windows.

Symbol The fonts for the symbol windows.

Popup The fonts for the popup window, the default window and the upper part of the define window.

If one of these fields is empty, the font for the edit window will be use for the related windows. By clicking on the description of a field, a popup menu with the available groups will appear where a group of fonts can be selected.

Keys The available keyboard commands can be changed by specifying a number of keyboard files. These files have the extension '.mpk', which will be added automatically. Multiple filenames are separated by colons (:).

Save Time The time between two periodic save operations can be specified here. A period will start when a edit action is performed and the save operation will only save those documents and stencils that are changed since the last save operation. The saved documents and stencil get the name *#name#*. The documents and stencils will also be saved in a dump file when the system crashes for some reason. The specified time can be between 1 and 60 minutes.

Left margin This value defines how much tab positions there should be between a displayed expression and the left margin. The size of a tab is defined by the 'tab size' fields.

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 in the middle of a character. By moving the cursor or resizing the window, the entire string can be viewed.

In the numerical field only digits can be entered while in a text field all characters can be entered. Beside the emacs commands from section 5.4 the **C-return** and **C-tab** can be used to move to the previous field and the **return** and **tab** can be used to move to the next field.