# MathSpad : a system for on-line preparation of mathematical documents 

Citation for published version (APA):

Backhouse, R. C., Verhoeven, R., \& Weber, O. (1995). MathSpad : a system for on-line preparation of mathematical documents. (Computing science reports; Vol. 9510). Technische Universiteit Eindhoven.

## Document status and date:

Published: 01/01/1995

## Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

## Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.
Link to publication


## General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25 fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

## Take down policy

If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

# Eindhoven University of Technology <br> Department of Mathematics and Computing Science 

## Math fpad: A System for On-Line Preparation of Mathematical Documents

by<br>R.C. Backhouse, R. Verhoeven and O. Weber

95/10

ISSN 0926-4515
All rights reserved
editors: prof.dr. J.C.M. Baeten
prof.dr. M. Rem

# Math/pad: A System for On-Line Preparation of Mathematical Documents 

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

March 1995

## 1 Introduction

Mathfpad 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 [5] and ATEX document-preparation systems [6] by providing a mechanism for converting documents prepared with the aid of the system into $\mathrm{IAT}_{\mathrm{E}}$ documents. Like these systems MathJpad emphasises logical design rather than visual design and is not a WYSIWYG (what you see is what you get) system. Unlike TEX and HTEX, however, MathJpad provides an ergonomically-designed user interface that allows one to create and manipulate mathematical formulae at a computer terminal in a form whose readability approaches the high standard of graphics-based WYSIWYG systems. The current document is itself a MathJpad document and has been written in its entirety directly on-screen. Instructions for how to obtain MathJpad and how to read this document at your computer terminal are included in the final section.

Mathfad is of potential value to those authors whose work typically contains lots of items of a mathematical nature, for example mathematical formulae like

[^0]$$
\sum_{0 \leq k<m}\left\lfloor\frac{n k+x}{m}\right\rfloor=d\left\lfloor\frac{x}{d}\right\rfloor+\frac{m-1}{2} n+\frac{d-m}{2}
$$
or computer programs (with or without comments of a mathematical nature):

```
program example;
var
k,m : integer ;
x,y : real;
begin
    {0\leqm}
    k:= 0;y:= 1;
    {Invariant: y= x
        Variant:m-k}
    while }k<>m\mathrm{ do begin
        k:=k+1 ; y := y*x
    end
    {y=\mp@subsup{x}{}{m}}
end.
```

If, however, one restricts oneself to standard mathematical notation such as taught up to, but not including, university level it is likely that a software package that has been tailor-made to respect such standards is more suited to one's purposes. MathJpad begins to be an attractive alternative when one wishes to deviate from accepted notation and invent one's own. It may be preferred, 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:


The most important component of the MathJpad system is a "stencil"-definition mechanism which allows one to design and use one's own notation - if necessary on the fly during the preparation of a document. Included in this mechanism is a method for specifying the $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ output that the notation should generate thus ensuring a high-quality printed version of the document even if the screen version has some shortcomings.

The use of Mathfpad 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 one belongs to a very small in-crowd of researchers one will not have any idea what this calculation is about. That is indeed an advantage since then viewing the calculation from a purely syntactic perspective becomes unavoidable. What one observes 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. One notes, for example, that the pattern " $f \circ$ " is repeated without change, and that the subexpression " $\{G F ; G . f\rceil$ ) is also copied repeatedly but in different positions in the formula.

$$
\begin{aligned}
& \beta \circ \alpha \\
& =\quad\{\quad \text { definitions of } \beta \text { and } \alpha \quad\} \\
& f \circ F .\left(G F ; G . f \emptyset \circ F . \text { in }_{G F}\right. \\
& =\{\quad F \text { is a functor }\} \\
& f \circ F \cdot\left(\mathbb{Q} G F ; G . f\left(\circ \operatorname{in}_{G F}\right)\right. \\
& =\quad\{\quad \text { SELF }\} \\
& f \circ F .(G . f \circ G . F .(G F ; G . f)) \\
& =\quad\{\quad G \text { is a functor }\} \\
& f \circ F . G .(f \circ F .(G F ; G . f D) \\
& =\quad\{\quad \text { definition of } \beta \quad\} \\
& f \circ F . G . \beta \text {. }
\end{aligned}
$$

MathJpad 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, Mathfpad has been designed as a hybrid between WYSIWYG and macrobased document preparation systems. It is intended to appeal to those whose work demands the comprehensiveness, power and flexibility of a logical-design system like TEX or ATEX. By choosing to use Matḩpad one will not be sacrificing any of these advantages. (Nor is it necessary to rewrite old documents: a Math/jad document can include any amount of raw $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ or $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ code. No special conversion program is necessary.) On the other hand, one gains the indispensable advantage of a visual design system so that one can read what one has written as one writes.

MathJpad inevitably has its limitations, some minor, the most major. 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 relatively primitive compared to conventional editing systems. Even for the input of mathematics its ergonomics is severely hampered by the inhuman nature of keyboard-mouse input, and
there is at present no mechanism to interface MathJpad with any of the many mathematical systems that are currently available. Mathßpad 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.

This paper comprises an overview of the main elements of the system.

## 2 Design Considerations

### 2.1 Background

Development of the MathJpad system was initiated in 1987 by Roland Backhouse. The priorities identified at that time have had a major influence on the design of Mathßpad and are therefore worth enumerating. A brief description of the research undertaken by Backhouse's group and our own perspective on the historical development of computerised mathematics also help to justify our choice of priorities.

Our research concerns the mathematics of program construction. That is, we are concerned with advancing the use of formal, mathematically-based program development systems. (By "system" we do not mean computer system in this context.) The articles we write tend to be highly mathematical in nature and they contain large quantities of algebraic calculations. More importantly, because our work is concerned with research into formal systems (with the emphasis on the final "s"), articles we write are varied and the mathematics they employ is non-standard.

Computer support for mathematical work has a long tradition and takes various forms - including algebraic manipulation systems like Maple and Mathematica, theorem provers like that designed and implemented by Boyer and Moore [1] and proof assistants like mural [4]. Typically, the goal of such systems is to improve the reliability of mathematical calculation. In the area of program development systems (of which mural and the BoyerMoore theorem prover are examples) the emphasis has traditionally been on the use of the computer as verifier. This tradition is not surprising given the history of computer development: true "interaction" with a computer is only now becoming possible and the use of a computer as verifier was until recently about the only feasible application imaginable. In the design of MathJpad, the goal from the outset was however to develop a system to support the creative activity of doing mathematics, in keeping with our research goal of developing mathematics of program construction (rather than verification).

The possibility of developing a system like MathJpad began to emerge in the late eighties. At the time of the project's initiation TEX had begun to firmly establish itself among the computing science community. The flexibility and high quality of $\mathrm{T}_{\mathrm{E}} \mathrm{X}$ output created a minor revolution in the preparation of scientific papers. In spite of the major investment needed to learn $\mathrm{T}_{\mathrm{E}} \mathrm{X}$, or its successor $\mathrm{BT}_{\mathrm{E}} \mathrm{X}$, and both systems subsequent unabated unfriendliness, many thousands of scientists now make daily use of them and would at no costs wish to relinquish their advantages for a more user-friendly - but inevitably more restrictive -- what-you-see-is-what-you-get system.

At about the same time the use of window systems began to exert a major influence on the design of computer software. Instead of the norm being what one might call "teletype systems" - systems characterised by monospaced input/output and an orderly commandline mode of operation - the idea that the computer screen might be made to resemble one's desk - characterised typically by apparently permanent disorder and a never empty pending list of assignments - became concrete.

With this background, a project was formulated with the goal of constructing an onscreen "scratch pad" for doing symbolic algebraic calculations. The resulting prototype system, the design of which was formulated in intensive discussions between Paul Chisholm and Roland Backhouse, was implemented by Chisholm with technical support from Harm Paas, and became available for general use in 1990 [2]. The principal design element of this system was a flexible mechanism for on-the-fly definition of mathematical notations (the "stencils" of the current system).

### 2.2 A Question of Priorities

Suppose that one's research goal is to enhance the reliability of mathematical calculation in some advancing field of mathematics and one sets out to implement a tool to support that goal. Bearing the limitations of current technology in mind, what should one's priorities be? This is a fundamental question that dictates the nature of the tool that is built.

The first priority set in the design of Math $\int$ pad was readability. Mathematical notation has been designed over centuries to be eminently readable, so a first requirement of the Mathfpad system was that one should be able to work directly with mathematical notation (and not some approximation to mathematical notation).

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

Typewriter font is also unreadable, although it has been around for so long that most of us do not realise the training that has to be undergone to learn how to read it. To illustrate the point consider the expression $a+b . c+d$ as written here in typewriter font. The justification for the mathematical convention that multiplication is denoted by the smallest symbol (a dot) and has the highest precedence is totally negated by the white space around the symbol created by the use of a non-proportionally-spaced font. The eye naturally groups the expression into the two subexpressions $a+b$ and $c+d$, and the intended structure of the expression is completely disguised. In general, the use of monospacing completely frustrates attempts to impart structure to the reader. It is indeed our contention that the continuing use of monospaced fonts as primary input medium to computer programs
is the most significant single source of programming error. The belief that one can do creative mathematics in such a font is as ridiculous as the belief that one can do creative mathematics in paper tape format.

Mathfpad documents (as displayed on-screen) are proportionally-spaced, allow multiple fonts (bold, sans serif, etc.), an unlimited symbol set (Greek letters: $\alpha, \beta$ etc., special symbols like (c), $\boldsymbol{\%}$, and so on), stacked expressions and subscripts and superscripts. See figs. 2-4 for illustrations. (All figures are collected together at the end of this document.)

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

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

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

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

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

## 3 System Highlights

### 3.1 Stencils

The most important element in the design of MathJpad is formed by the so-called stencils. A MathJpad stencil is a file describing the notational conventions used in one or more MathJpad documents, and how these are to be set using the $\mathrm{AA} \mathrm{T}_{\mathrm{E}} \mathrm{X}$ document preparation system. Each stencil consists of a number of templates, each of which may have various versions.

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 [3]. and incorporated them in the stencil "Gries-Schneider". This stencil, as it appears on the screen is shown in fig. 1.

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 a $\mathrm{ET}_{\mathrm{EX}}$ stencil, in addition to the Gries-Schneider stencil.)

Figure 2 shows an extract from one page of their book (p.151); fig. 2(a) is a screen dump, and fig. 2(b) shows the extract after processing with $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$.

The Gries-Schneider stencil (fig. 1) illustrates the general form of a Mathfpad stencil. There are six sections, labelled None, Prefix, Postfix, Infix, 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 Gries-Schneider stencil - reading from top to bottom, $ᄀ, \equiv, \wedge,=, ., \Leftarrow$ and $\Rightarrow$. 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 one moves the mouse to point at " $\wedge$ " in the stencil and presses down with the rightmost mouse button a small menu will appear containing two items " $\wedge$ " and " $V$ ". The two operators have different meanings, of course, but according to the definition in the stencil, they are treated by Math $\int \mathrm{j}$ pad as different "versions" of the same operator. In fact the total list of operators defined in the stencil amounts to sixteen.

The operator " $[\mathbf{V}:=\mathrm{E}]$ " is somewhat unconventional although its use should be evident from fig. 2. It illustrates a common phenomenon in mathematics, namely the use of an operator that is parameterised, in this case by a variable "V" and an expression "E". The operator "O $\langle\mathbf{T}\rangle$ " in the stencil is also parameterised by an operator "O" and text "T". It has been used to create the proof steps in fig. 2. Its use will be discussed further later.

Each operator has been assigned a precedence according to a list to be found at the beginning of Gries and Schneider's book. 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. By clicking on entries in the stencil expressions can be entered into a Mathऽpad edit document. This can be done either top-down ("syntaxdirected"), or from left to right, or indeed using a combination of top-down and left to right input.

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 example, a name - "Change of dummy" in the example -, a condition or context under which the rule is valid - "Provided $\rightarrow \operatorname{occurs}\left({ }^{\prime} y^{\prime}, ' R, P^{\prime}\right.$ ) and $f$ has an inverse" in the 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 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 target selection:

## (T) Name: T: <br> Condition: $\mathbf{T}$, Statement: E

The three bold T's and the bold $\mathbf{E}$ 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 ATEX output generated by the MathJpad document.
(Compare fig. 2(a) with fig. 2(b).) The Math/pad system incorporates an automatic mechanism for generating $\mathrm{ETEX}_{\mathrm{E}}$ output so that in principle it is unnecessary for the user to learn $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ in order to use the system. In practice it is sometimes desirable to over-ride the system-generated $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ either, as in this case, because the screen version of a template is designed to assist in the writing of the document, or to improve the quality of the output. The general design principle that this illustrates is: the user is the boss and can always over-ride automatic actions of the system.

The creation of (versions of) a template in a stencil can be very straightforward; it can also be the most difficult task of all in the use of Math/pad. It is very straightforward to enter simple mathematical operators. The difficulties begin when you require special layout conventions. The difficulties are compounded when the $\mathrm{AT}_{\mathrm{E}} \mathrm{X}$ automatically generated by MathJpad is not adequate to your needs. This is undoubtedly a major hurdle for the beginning user of MathJpad. The experienced user will, however, already have several stencils at their disposal and will seldom need to augment or modify them. When it is necessary to do so this can always be done "on-the-fly". Moreover, if a sufficiently large community of Mathjpad users develops, we expect (hope!) that in time stencils for particular application areas (a C stencil, a category theory stencil, a calculus stencil etc.) will become readily available and will be shared among users thus making the initial hurdle for beginners much more palatable.

### 3.2 Expression Manipulation

The primary goal of MathJpad 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 quịckly whereas human beings tend to introduce mistakes in the process.

The fact that MathJpad is a structure editor pays enormous dividends in the ease with which substructures may be selected. To select an expression in a text it suffices to point the mouse at the principal operator in the expression and click. In other systems that we know of selection is effected by dragging the mouse -- usually over some encoding of the expression involving "funny" control characters or umpteen backslashes and macro names rather than the expression itself. The result is that the most basic editing operation, namely copying, becomes clumsy and hence unreliable in those systems. With MathJpad that is not the case.

Apart from being able to copy subexpressions easily and quickly, MathJpad provides simple-to-use functions for rearranging expressions, and for finding and replacing mathematical expressions of a given structure. The functions provided by Math/pad for expression manipulation are Copy and Swap, Reverse, Distribute and Factorise, Group and Ungroup, Apply, Rename and Find. The functions Swap, Copy and Find can also be used on plain text, but the remainder are peculiar to the manipulation of mathematical expressions.

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 and not to
be confused with evaluation of the function.) Rename is used to rename dummy variables within a given scope. We do not discuss these operations any further here.

Find is short for Find / Find-and-Replace. It can be used to search for, and possibly replace, both text and mathematical expressions. The effectiveness of Find is considerably enhanced by the fact that Math/pad is a structure editor. It can be used to search for expressions of a particular shape simply by specifying a pattern containing one or more unfilled place-holders. By combining this with a replacement pattern the operation can be used to transform expressions according to mathematical rules. For example, Find-andReplace can be used to transform all (or some) expressions of the shape (E1+E2)* to $\mathbf{E 1 *} \circ\left(\mathbf{E} 2 \circ \mathbf{E} 1^{*}\right)^{*}$ where $\mathbf{E} 1$ and $\mathbf{E} 2$ are place holders for expressions.

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 MathJpad are geared to the use of infix notation in the two ways explained above. When creating a MathJpad 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 MathJpad 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 $a \Rightarrow(b \Rightarrow c)$. )

The Matḩad functions that exploit the notion of associativity are Reverse, Distribute and its inverse Factorise. The function Reverse reverses the order of the elements in a
list of arguments and (associative) infix operators. For example, reversing the expression $a \leq b<c=d \leq e$ results in the expression $e \leq d=c<b \leq a$. (Note that the operation does not replace " $<$ " by " $>$ " or " $\leq$ " by " $\geq$ ". ) Subexpressions can also be reversed. Reversing the subexpression $(c \div d) \cdot e$ in $a \cdot b+(c \div d) \cdot e+f$, for example, results in $a \cdot b+e \cdot(c \div d)+f$. (Note that the operation is not recursive: the arguments of the division operator are not reversed.) Distribute and Factorise 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 ease with which one can select, copy and rearrange expressions using these few simple operations makes MathJpad a very "handy" symbolic calculator, particularly in areas of mathematics where no specialised machine support is available. Yet more benefit can be obtained from the system if one is prepared to use it in seemingly unorthodox ways. The use of the template labelled " $\mathrm{O}\langle\mathbf{T}\rangle$ " in the Gries-Schneider stencil is one such example. It defines a binary associative operator, just like the addition operator, except that the operator itself has two arguments: "O", which is used for the relation holding between the operator's two arguments in a proof step, and " T ", which is used for the explanation of the proof step. (Note that the explanation can run over several lines: see fig. 2.) In this way no special "feature" has to be added to the system in order to accommodate the calculational proof style (or indeed other proof styles) used by Gries and Schneider in their book.

## 4 The Current Status

The first alpha release of the MathJpad system took place in December 1993, and a second release is due to take place in the spring of 1995 . The system is freely available via world-wide web at http://www.win.tue.nl/win/cs/wp/mathspad. Several documents produced with the aid of the system (almost all documents inserted in the directory in 1993 or later) are stored at http://www.win.tue.nl/win/cs/wp/papers.

The MathJpad manual is itself a MathJpad document and, with the exception of screen dumps, is distributed with the system. It is planned to publish the manual in book form in the near future.

## Acknowledgement

Figures 3 and 4 are extracted from unpublished work by Kevin Backhouse and Paul Hoogendijk, respectively. We are very grateful to them for allowing us to include them in this document.


Figure 1: The Gries-Schneider stencil

## References

[1] R.S. Boyer and J S. Moore. A Computational Logic. Academic Press, 1979.
[2] P. Chisholm. Calculation by computer. In Third International Workshop Software Engineering and its Applications, pages 713-728, Toulouse, France, December 3-7 1990. EC2.
[3] David Gries and Fred B. Schneider. A Logical Approach to Discrete Math. SpringerVerlag, 1993.
[4] C.B. Jones, K.D. Jones, P.A.Lindsay, and R. Moore. mural: A Formal Development Support System. Springer-Verlag, London, 1991.

[6] Leslie Lamport. ${ }^{A} T_{E} X$ : A Document Preparation System. Addison-Wesley, Reading, Massachusetts, 1986.

## (.8.22.) Name -Change of dummy ':

Condition: $\quad$ Provided $\neg$ occurs ( $\left.y^{\prime},{ }^{\prime} R, P^{p}\right)$ and $f$ has an inverse',
Statement: $\quad(* x \mid R: P)=(* y \mid E[x:=f, y]: P[x:=f, y])$
The proof of this theorem illustrates the use of several of the axioms given above.
The proof starts with the RHS of (8.22), because it has more structure.

$$
\begin{aligned}
& (\star y \mid E[x:=f, y]: P[x:=f, y]) \\
& =\text { (One point rule (8.14) } \\
& \text {---Quantification over } x \text { has to be introduced. The One- } \\
& \text { point rule is the oniy } \cdot \text { rule that can be applied at first. } \cdot \text { ) } \\
& (* y \mid E[x:=f, y]:(* x \mid x=f, y: P)) \\
& =\quad(\cdot \text { Nesting }(8.20)-- \text { Moving dummy } x \text { to the outside } \\
& \text { gets us closer to the final form. } \cdot \text { ) } \\
& (* x, y \mid R[x:=f, y] \wedge x=f, y: P)
\end{aligned}
$$

(a) Snapshot of Mathfpad document
(8.22) Change of dummy:

Provided $\neg$ occurs (' $\left.y^{\prime}, ~ ‘ ~ R, ~ P '\right) ~ a n d ~ f o s ~ a n ~ i n v e r s e, ~$
$(* x \mid R: P)=(* y \mid R[x:=f . y]: P[x:=f . y])$
The proof of this theorem illustrates the use of several of the axioms given above. The proof starts with the RHS of (8.22), because it has more structure.

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

(b) Output produced after processing with $\mathrm{Br}_{\mathrm{E}} \mathrm{X}$

Figure 2: Running example. (Note the integration of prose with mathematical calculations)
$\backslash$ font $\backslash$ pics $=$ kevpics scaled 1000
\unitlength 1 cm
\begin\{picture\}(0,0) }
$\backslash \operatorname{def} \backslash m y p u t(\# 1, \# 2) \# 3\{\backslash \operatorname{put}(\# 1, \# 2) \backslash \backslash$ makebox $(0,0)\{\# 3\}\}\}$
$\backslash$ myput(10.3,-1.7) $\backslash$ pics B$\}$
$\backslash$ myput $(10.3,-0.5)\left\{F_{\mathrm{n}}\right\}$
$\backslash$ myput $10,-2.0)\left\{F_{\text {rem }}\right\}$
$\backslash$ myput(10.8,-3.0)\{\vbox to $0 \mathrm{~cm}\left\{\backslash h b o x\right.$ to $\left.\left.0 \mathrm{~cm}\left\{\backslash \mathrm{hss} F_{\mathrm{z}} \backslash \mathrm{hss}\right\} \backslash \mathrm{vss}\right\}\right\}$
\end\{picture\}\% }
Volgens de regel hierboven geldt:

$$
\begin{aligned}
& F_{\mathrm{n}}=F_{\mathrm{z}} \cdot \frac{1}{\sqrt{1+\mathrm{c}_{1}^{2}}}=\frac{m \cdot g}{\sqrt{1+\left[\mathrm{f}^{\prime}(x)\right]^{2}}} \\
& F_{\mathrm{rem}}=F_{\mathrm{z}} \cdot \frac{c_{1}}{\sqrt{1+\mathrm{c}_{1}^{2}}}=\frac{m \cdot g \cdot \mathrm{f}^{\prime}(x)}{\sqrt{1+\left[\mathrm{f}^{\prime}(x)\right]^{2}}}
\end{aligned}
$$

(a) Snapshot of a Mathfpad document

Volgens de regel hierboven geldt:

$$
\begin{array}{ll}
F_{\mathrm{n}}=F_{\mathrm{z}} \cdot \frac{1}{\sqrt{1+c_{1}^{2}}}=\frac{m \cdot g}{\sqrt{1+\left[f^{\prime}(x)\right]^{2}}} \\
F_{\mathrm{rem}}=F_{\mathrm{z}} \cdot \frac{c_{1}}{\sqrt{1+c_{1}^{2}}}=\frac{m \cdot g \cdot f^{\prime}(x)}{\sqrt{1+\left[f^{\prime}(x)\right]^{2}}}
\end{array}
$$

(b) Output produced after processing with IATEX

Figure 3: Raw TEX may be included in Mathßad files.
(where rid $A: A \leftarrow A \times \mathbb{1}$ is the obvious natural isomorphism) commutes, and Diagram:

$$
\begin{array}{lll}
F(A \times(B \times C)) . & \stackrel{\theta_{A, B \times C}}{ } & F A \times(B \times C) \\
F a s s_{A, B, C} \downarrow . & \cdot & \downarrow a s s_{F A, B, C} \\
F((A \times B) \times C) \ll \theta_{A \times B, C} & F(A \times B) \times C \underset{\theta_{A, B} \times i d_{C}}{<} & (F A \times B) \times C
\end{array}
$$

(where ass $A, B, C:(A \times B) \times C \leftarrow A \times(B \times C)$ is the obvious natural isomorphism) commutes as well. A functor that has at least one strength is said to be strong.
(a) Snapshot of a MathJpad document
(where rid $_{A}: A \leftarrow A \times 1$ is the obvious natural isomorphism) commutes, and

(where ass $A_{A}, C$ : $(A \times B) \times C \leftarrow A \times(B \times C)$ is the obvious natural isomorphism) commutes as well. A functor that has at least one strength is said to be strong.
(b) Output produced after processing with IATEX

Figure 4: An example of using MathJpad for making diagrams using Paul Taylor's diagram macro package.

## Computing Science Reports

In this series appeared:

| 93/01 | R. van Geldrop | Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36. |
| :---: | :---: | :---: |
| 93/02 | T. Verhoeff | A continuous version of the Prisoner's Dilemma, p. 17 |
| 93/03 | T. Verhoeff | Quicksort for linked lists, p. 8. |
| 93/04 | E.H.L. Aarts J.H.M. Korst P.J. Zwietering | Deterministic and randomized local search, p. 78. |
| 93/05 | J.C.M. Baeten C. Verhoef | A congruence theorem for structured operational semantics with predicates, p. 18. |
| 93/06 | J.P. Veltkamp | On the unavoidability of metastable behaviour, p. 29 |
| 93/07 | P.D. Moerland | Exercises in Multiprogramming, p. 97 |
| 93/08 | J. Verhoosel | A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32. |
| 93/09 | K.M. van Hee | Systems Engineering: a Formal Approach Part I: System Concepts, p. 72. |
| 93/10 | K.M. van Hee | Systems Engineering: a Formal Approach Part II: Frameworks, p. 44. |
| 93/11 | K.M. van Hee | Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101. |
| 93/12 | K.M. van Hee | Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63. |
| 93/13 | K.M. van Hee | Systems Engineering: a Formal Approach Part V: Specification Language, p. 89. |
| 93/14 | J.C.M. Baeten <br> J.A. Bergstra | On Sequential Composition, Action Prefixes and Process Prefix, p. 21. |
| 93/15 | J.C.M. Baeten <br> J.A. Bergstra R.N. Bol | A Real-Time Process Logic, p. 31. |
| 93/16 | H. Schepers <br> J. Hooman | A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27 |
| 93/17 | D. Alstein <br> P. van der Stok | Hard Real-Time Reliable Multicast in the DEDOS system, p. 19. |
| 93/18 | C. Verhoef | A congruence theorem for structured operational semantics with predicates and negative premises, p. 22. |
| 93/19 | G-J. Houben | The Design of an Online Help Facility for ExSpect, p.21. |
| 93/20 | F.S. de Boer | A Process Algebra of Concurrent Constraint Programming, p. 15. |


| 93/21 | M. Codish <br> D. Dams <br> G. Filé <br> M. Bruynooghe | Freeness Analysis for Logic Programs - And Correctness?, p. 24. |
| :---: | :---: | :---: |
| 93/22 | E. Poll | A Typechecker for Bijective Pure Type Systems, p. 28. |
| 93/23 | E. de Kogel | Relational Algebra and Equational Proofs, p. 23. |
| 93/24 | E. Poll and Paula Severi | Pure Type Systems with Definitions, p. 38. |
| 93/25 | H. Schepers and R. Gerth | A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31. |
| 93/26 | W.M.P. van der Aalst | Multi-dimensional Petri nets, p. 25. |
| 93/27 | T. Kloks and D. Kratsch | Finding all minimal separators of a graph, p. 11. |
| 93/28 | F. Kamareddine and R. Nederpelt | A Semantics for a fine $\lambda$-calculus with de Bruijn indices, p. 49. |
| 93/29 | R. Post and P. De Bra | GOLD, a Graph Oriented Language for Databases, p. 42. |
| 93/30 | J. Deogun <br> T. Kloks <br> D. Kratsch <br> H. Müller | On Vertex Ranking for Permutation and Other Graphs, p. 11. |
| 93/31 | W. Körver | Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40. |
| 93/32 | H. ten Eikelder and H. van Geldrop | On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17. |
| 93/33 | L. Loyens and J. Moonen | ILIAS, a sequential language for parallel matrix computations, p. 20. |
| 93/34 | J.C.M. Baeten and J.A. Bergstra | Real Time Process Algebra with Infinitesimals, p.39. |
| 93/35 | W. Ferrer and P. Severi | Abstract Reduction and Topology, p. 28. |
| 93/36 | J.C.M. Baeten and <br> J.A. Bergstra | Non Interleaving Process Algebra, p. 17. |
| 93/37 | J. Brunekreef J-P. Katoen R. Koymans <br> S. Mauw | Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73. |
| 93/38 | C. Verhoef | A general conservative extension theorem in process algebra, p. 17. |
| 93/39 | W.P.M. Nuijten <br> E.H.L. Aarts <br> D.A.A. van Erp Taalman Kip <br> K.M. van Hee | Job Shop Scheduling by Constraint Satisfaction, p. 22. |


| 93/40 | P.D.V. van der Stok M.M.M.P.J. Claessen <br> D. Alstein | A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43. |
| :---: | :---: | :---: |
| 93/41 | A. Bijlsma | Temporal operators viewed as predicate transformers, p. 11. |
| 93/42 | P.M.P. Rambags | Automatic Verification of Regular Protocols in P/T Nets, p. 23 |
| 93/43 | B.W. Watson | A taxomomy of finite automata construction algorithms, p. 87. |
| 93/44 | B.W. Watson | A taxonomy of finite automata minimization algorithms, p. 23. |
| 93/45 | E.J. Luit <br> J.M.M. Martin | A precise clock synchronization protocol,p. |
| 93/46 | T. Kloks <br> D. Kratsch <br> J. Spinrad | Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14. |
| 93/47 | W. v.d. Aalst P. De Bra G.J. Houben Y. Komatzky | Browsing Semantics in the "Tower" Model, p. 19. |
| 93/48 | R. Gerth | Verifying Sequentially Consistent Memory using Interface Refinement, p. 20. |
| 94/01 | P. America <br> M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart | The object-oriented paradigm, p. 28. |
| 94/02 | F. Kamareddine R.P. Nederpelt | Canonical typing and $\Pi$-conversion, p. 51. |
| 94/03 | L.B. Hartman K.M. van Hee | Application of Marcov Decision Processe to Search Problems, p. 21. |
| 94/04 | J.C.M. Baeten <br> J.A. Bergstra | Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18. |
| 94/05 | P. Zhou <br> J. Hooman | Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22. |
| 94/06 | T. Basten <br> T. Kunz <br> J. Black <br> M. Coffin <br> D. Taylor | Time and the Order of Abstract Events in Distributed Computations, p. 29. |
| 94/07 | K.R. Apt <br> R. Bol | Logic Programming and Negation: A Survey, p. 62. |
| 94/08 | O.S. van Roosmalen | A Hierarcḥical Diagrammatic Representation of Class Structure, p. 22. |


| 94/09 | J.C.M. Baeten <br> J.A. Bergstra | Process Algebra with Partial Choice, p. 16. |
| :---: | :---: | :---: |
| 94/10 | T. verhoeff | The testing Paradigm Applied to Network Structure. p. 31 . |
| 94/11 | J. Peleska <br> C. Huizing <br> C. Petersohn | A Comparison of Ward \& Mellor's Transformation Schema with State- \& Activitycharts, p. 30. |
| 94/12 | T. Kloks <br> D. Kratsch <br> H. Müller | Dominoes, p. 14. |
| 94/13 | R. Seljée | A New Method for Integrity Constraint checking in Deductive Databases, p. 34. |
| 94/14 | W. Peremans | Ups and Downs of Type Theory, p. 9. |
| 94/15 | R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra | Job Shop Scheduling by Local Search, p. 21. |
| 94/16 | R.C. Backhouse H. Doombos | Mathematical Induction Made Calculational, p. 36. |
| 94/17 | S. Mauw M.A. Reniers | An Algebraic Semantics of Basic Message Sequence Charts, p. 9. |
| 94/18 | F. Kamareddine R. Nederpelt | Refining Reduction in the Lambda Calculus, p. 15. |
| 94/19 | B.W. Watson | The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46. |
| 94/20 | R. Bloo <br> F. Kamareddine <br> R. Nederpelt | Beyond $\beta$-Reduction in Church's $\lambda \rightarrow$, p. 22. |
| 94/21 | B.W. Watson | An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions. |
| 94/22 | B.W. Watson | The design and implementation of the FIRE engine: <br> A C++ toolkit for Finite automata and regular Expressions. |
| 94/23 | S. Mauw and M.A. Reniers | An algebraic semantics of Message Sequence Charts, p. 43. |
| 94/24 | D. Dams <br> O. Grumberg <br> R. Gerth | Abstract Interpretation of Reactive Systems: <br> Abstractions Preserving $\forall C T L^{*}, \exists C T L *$ and CTL*, p. 28. |
| 94/25 | T. Kloks | $\mathrm{K}_{1,3}$-free and $\mathrm{W}_{4}$-free graphs, p. 10. |
| 94/26 | R.R. Hoogerwoord | On the foundations of functional programming: a programmer's point of view, p. 54 . |
| 94/27 | S. Mauw and H. Mulder | Regularity of BPA-Systems is Decidable, p. 14. |
| 94/28 | C.W.A.M. van Overveld M. Verhoeven | Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20. |


| 94/29 | J. Hooman |  | Correctness of Real Time Systems by Construction, p. 22. |
| :---: | :---: | :---: | :---: |
| 94/30 | J.C.M. Baeten <br> J.A. Bergstra <br> Gh. Ştefanescu |  | Process Algebra with Feedback, p. 22. |
| 94/31 | B.W. Watson R.E. Watson |  | A Boyer-Moore type algorithm for regular expression pattern matching, p. 22. |
| 94/32 | J.J. Vereijken |  | Fischer's Protocol in Timed Process Algebra, p. 38. |
| 94/33 | T. Laan |  | A formalization of the Ramified Type Theory, p.40. |
| 94/34 | R. Bloo <br> F. Kamareddine <br> R. Nederpelt |  | The Barendregt Cube with Definitions and Generalised Reduction, p. 37. |
| 94/35 | J.C.M. Baeten S. Mauw |  | Delayed choice: an operator for joining Message Sequence Charts, p. 15. |
| 94/36 | F. Kamareddine R. Nederpelt |  | Canonical typing and $\Pi$-conversion in the Barendregt Cube, p. 19. |
| 94/37 | T. Basten <br> R. Bol <br> M. Voorhoeve |  | Simulating and Analyzing Railway Interlockings in ExSpect, p. 30. |
| 94/38 | A. Bijlsma C.S. Scholten |  | Point-free substitution, p. 10. |
| 94/39 | A. Blokhuis T. Kloks |  | On the equivalence covering number of splitgraphs, p. 4. |
| 94/40 | D. Alstein |  | Distributed Consensus and Hard Real-Time Systems, p. 34. |
| 94/41 | T. Kloks D. Kratsch |  | Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6. |
| 94/42 | J. Engelfriet J.J. Vereijken |  | Concatenation of Graphs, p. 7. |
| 94/43 | R.C. Backhouse M. Bijsterveld |  | Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35. |
| 94/44 | E. Brinksma <br> R. Gerth S. Graf <br> W. Janssen <br> S. Katz <br> M. Poel <br> C. Rump | J. Davies <br> B. Jonsson <br> G. Lowe <br> A. Pnueli <br> J. Zwiers | Verifying Sequentially Consistent Memory, p. 160 |
| 94/45 | G.J. Houben |  | Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43. |
| 94/46 | R. Bloo <br> F. Kamareddine <br> R. Nederpelt |  | The $\lambda$-cube with classes of terms modulo conversion, p. 16. |


| 94/47 | R. Bloo <br> F. Kamareddine R. Nederpelt | On II-conversion in Type Theory, p. 12. |
| :---: | :---: | :---: |
| 94/48 | Mathematics of Program Construction Group | Fixed-Point Calculus, p. 11. |
| 94/49 | J.C.M. Baeten <br> J.A. Bergstra | Process Algebra with Propositional Signals, p. 25. |
| 94/50 | H. Geuvers | A short and flexible proof of Strong Normalazation for the Calculus of Constructions, p. 27. |
| 94/51 | T. Kloks <br> D. Kratsch <br> H. Müller | Listing simplicial vertices and recognizing diamond-free graphs, p. 4. |
| 94/52 | W. Penczek <br> R. Kuiper | Traces and Logic, p. 81 |
| 94/53 | R. Gerth <br> R. Kuiper <br> D. Peled <br> W. Penczek | A Partial Order Approach to Branching Time Logic Model Checking, p. 20. |
| 95/01 | J.J. Lukkien | The Construction of a small CommunicationLibrary, p.16. |
| 95/02 | M. Bezem <br> R. Bol <br> J.F. Groote | Formalizing Process Algebraic Verifications in the Calculus of Constructions, p. 49. |
| 95/03 | J.C.M. Baeten C. Verhoef | Concrete process algebra, p. 134. |
| 95/04 |  |  |
| 95/05 | P. Severi | A Type Inference Algorithm for Pure Type Systems, p. 20. |
| 95/06 | T.W.M. Vossen M.G.A. Vehoeven H.M.M. ten Eikelder E.H.L. Aarts | A Quantitative Analysis of Iterated Local Search, p.23. |
| 95/07 | G.A.M. de Bruyn O.S. van Roosmalen | Drawing Execution Graphs by Parsing, p. 10. |
| 95/08 | R. Bloo | Preservation of Strong Normalisation for Explicit Substitution, p. 12. |
| 95/09 | J.C.M. Baeten <br> J.A. Bergstra | Discrete Time Process Algebra, p. 20 |


[^0]:    *Now at CWI, Amsterdam

