Riaca's Interactive Algebra

Roderik Lindenbergh,
Universiteit Utrecht,
Department of Mathematics,
P.O.box 80010,
3508 TA Utrecht,
The Netherlands

lindenbe@math.ruu.nl

June, 1996

Contents

1 Preface

This technical report describes the interactive algebra book project I have been working on, together with other people, in the period April 1995 - March 1996. It will give the setup of the project, describe the achievements after one year and will provide the reader with a manual on "How to create my own interactive mathematics book?". At the end, it will list up some ideas for the future.
I was able to work on this project for one year because I had to fulfill my alternative military service. The work was done at Riaca and at the Technical University of Eindhoven under the supervision of Professor Arjeh Cohen, who was also the initiator of this project. Here I would like to express my thanks to Dr. Hans Cuypers and Dr. Hans Sterk who, together with Arjeh Cohen are in charge of the course material that serves as a source for the book. I would like to thank Dr. Garth Dickie as well, who is continuing my work. Special thanks go to a number of inspirators, that where around to work with and to ask questions to: Simon Atkins, André Heck, Dr. Leendert van Gastel, Dr. Marc van Leeuwen and Dr. Dima Pasechnik between others. Last but not least I would like to thank Arjeh Cohen who made it possible for me to work on this nice project for one year and who was giving me loads of good ideas and suggestions!

2 An interactive algebra book

2.1 Aim of the project

During a period of one year we have been working on an interactive algebra book project. The aim of this project was to create an interactive algebra book in one year using existing software. Meanwhile problems with and shortcomings of the software should be noted and give the developers ideas of how to create better versions in the future. After one year the interactive book should be in such a state that demo versions could be presented to students. Reactions of students should also help in getting ideas how to proceed with interactive books.

2.2 Requirements

A LaTeX-source text for the algebra lecture notes was already existing. For that reason it would be nice if this text could also serve as source for the interactive versions of the book. Meanwhile, tools should be developed that enable staff to create interactive books from LaTeX-source in general, without the necessity of knowing a lot about the used software and formats. As the staff in Eindhoven works on Unix machines and as most students use PC's that run Windows 3.1, these kind of operating systems should be supported. As an algebra text contains a lot of mathematical notation, formats should be used that handle mathematical texts in an adequate way. It was clear that Computer Algebra Systems like Maple, Mathematica and Gap should play an important role.

2.3 Kinds of interactivity

As features that make the book interactive one could think of

- hyperlinks inside the text.

- A text presented in layers.
- Mathematica Notebooks, Maple Worksheets, ... , connected to the text via links.
- Stand alone applications, especially developed to make the user familiar with certain mathematical concepts, connected to the text via links.

These features are all present in the resulting interactive algebra book, and will be described in more detail in the following paragraphs.

2.4 Choosing formats

The first month of the project was dedicated to the inventarisation of existing software. Only a few formats are able to display mathematical texts in a nice way: some front-ends to Computer Algebra Systems do, like the X version of Mathematica or Maple. The main problem with these is that they restrict the user/developer to one Computer Algebra System on forehand. Besides, these products are commercial and it is therefore very difficult to manipulate them as the source code is not freely available. Some word-processors like Scientific Word do, but they are commercial as well and not present on Unix. HTML, the format used by most World Wide Web browsers only supports very few mathematical symbols: in typesetting mathematical texts on the World Wide Web it was common practice to convert pieces of mathematics in small pictures in GIF format and display these pictures in the browser. Unfortunately the quality of these pictures is not very good.

2.5 The chosen format: a TEX-family

A well-known program that is used to display mathematical typesetted texts is XDVI. DVI is a format that is obtained when one compiles TEX– or LATEX–source. It turned out that a special version of XDVI, called XHDVI also supports hyperlinks. As it is possible to launch external applications from within XHDVI as well, this was a good program to start with. Besides, a new format showed up, PDF, that can be obtained by converting Postscript. Readers for documents in PDF formats are available for most operating systems now, including Windows 3.1. The combination of DVI and PDF, and the route TEX $\rightarrow$ DVI $\rightarrow$ Postscript $\rightarrow$ PDF enables us to work with one single source document that will be converted in different formats that will run on different platforms. Fortunately, TEX is so general that it will keep suiting our purposes, even when other formats and programs will become available.

3 The global setup

The following sections will describe a collection of LATEX commands, TEX macros, format converters and scripts that enable you to turn a LATEX source document into (interactive) course material. More specifically, with the described packages you can create a paper book, a book that runs on UNIX and a book that will run on a PC operating with Windows 3.1. Firstly I will describe how to create suitable LATEX source. After that I will tell how to create the three different books. While doing so I will treat the software we used. Computer Algebra Systems like Mathematica and stand alone applications can be linked to the computer books. I will tell how to achieve this and will describe some of the stand alone applications. Finally

I will list up known problems and ideas that could be implemented in the future.

The source document is divided into chapters. Every chapter will live in its own file. Different books origin from different root files that will combine all the chapters and will choose the appropriate macro. Table 1 gives an overview of the versions and macros used:

End product	paper	Unix/XHDVI	Windows3.1/Acrobat Reader
Root file	`root.tex`	`root_dvi.tex`	`root_win.tex`
Macro name	`textbook.tex`	`hypbook.tex`	`winhyp.tex`
Additional macro's	— `hyphalg.tex`	`hypdet.tex` `hyphalg.tex`	`hypdet.tex` `hyphalg.tex`

Table 1: Different books

In the following I will indicate both the Unix and Windows book with *computer book*. The author needs only to edit the root file for adding or removing chapters, changing titles etc. The macro file **`hyphalg.tex`** contains hyphenation patterns. All other macros are used to interpret commands in the source document in a proper way.

4 The LaTeX commands

4.1 Layers and Units

New LaTeX commands are introduced to enlarge the (interactive) possibilities of the course material. Their functionality will depend on the kind of book. This section will describe these new commands. To create layers, environments are introduced in the source document that enable a script to take out parts of text while parsing the source. In all books, the course material will be divided into units providing the users with small blocks of information. In the text book the units are at lowest level in the hierarchy; in the computer book one descends further into details and other kinds of sub-item. The level of the units will be called main level, as it should give all basic information.

Everything that should not belong to the main level should be put between **\begin{<type>}...** and a **\end{<type>}**. Of course it is also necessary to indicate what should not be printed in the text book, see Table 2.

destination	type
detail file	detail
Mathematica files	matica
Maple files	maple
solutions file	sol
do not print	noprint

Table 2: different file types

Now the author has to use the first script, called splits. In case of one of the first three types in Table 2 the splits script will copy everything between the \begin{<type>} mark and the \end{<type>} mark to the file determined by type, and will put a \go{<type>}... command on the corresponding place in the main level file. In the detail case, the \begin{<type>} mark and the \end{<type>} mark and the text in between, will be copied most of the time in the same order to the detail file. Sometimes (in the case of another level) however, this is not possible and in that case splits will sort her input first. The contents of the Mathematica files will be converted to Mathematica worksheets. Everything in between \begin{noprint} and \end{noprint} will be ignored by LaTeX in the text book.

4.2 Starting chapters and sections

```
\begin{chaphead}{#1} <abstract> \end{chaphead}
```

Works as the usual LaTeX command \chapter{#1}. #1 is the title of the chapter; the text inside the environment should give a short summary of the contents of that chapter.

```
\startsection{#1}
```

Works as the usual LaTeX command \section{#1}.

4.3 Starting units

Every section will be divided into units. The start of a new unit in the text book will be marked by the appearance of a bold unit number. Several special types exist. They are listed in table 3. In the text they will be recognized because their type name will appear in the unit header. When a unit does not belong to the types mentioned in the table, it should start with

```
\+{#1}
```

This will result in a unit header that only contains a number. When some text is needed too, one should use the command

```
\st{#1}.
```

Currently, these unit starters are available:

start with	end with	text book:	computer book:
`\+{#1}`			**#1**
`\begin{al}{#1}`	`\end{al}`	**Algoritme.**	**Algoritme: #1**
`\begin{app}{#1}`	`\end{app}`	**Algoritme.**	**Toepassing: #1**
`\begin{co}{#1}`	`\end{co}`	**Constructie.**	**Constructie: #1**
`\begin{cor}{#1}`	`\end{cor}`	**Corollarium.**	**Corollarium: #1**
`\begin{df}{#1}`	`\end{df}`	**Definitie.**	**Definitie: #1**
`\begin{dfs}{#1}`	`\end{dfs}`	**Definities.**	**Definities: #1**
`\begin{ec}{#1}`	`\end{ec}`	**Opgave.**	**Opgave: #1**
`\begin{ex}{#1}`	`\end{ex}`	**Voorbeeld.**	**Voorbeeld: #1**
`\begin{exs}{#1}`	`\end{exs}`	**Voorbeelden.**	**Voorbeelden: #1**
`\begin{lem}{#1}`	`\end{lem}`	**Lemma.**	**Lemma: #1**
`\begin{prop}{#1}`	`\end{prop}`	**Propositie.**	**Propositie: #1**
`\begin{rmk}{#1}`	`\end{rmk}`	**Opmerking.**	**Opmerking: #1**
`\begin{rmks}{#1}`	`\end{rmks}`	**Opmerkingen.**	**Opmerkingen: #1**
`\st{#1}`		**#1**	**#1**
`\begin{thm}{#1}`	`\end{thm}`	**Stelling.**	**Stelling: #1**

Table 3: Unit starters

4.4 Labels and cross references

When the author defines a label just after having started an unit, the cross reference will display the type of the unit (if present) together with the number of the unit (always present). In the computer books these references will be click-able. The script `splits` and the macro files will change references to Mathematica parts in the paper book into links to the corresponding Mathematica Notebook in the UNIX book and in a inactive link in the case of the Windows book.

4.5 Units in the text book; details in the computer book

A detail in the computer book is living in a detail file but has always a parent (c.q. a unit) in the main level file. To enable communication between parents and children (c.q. units and details) it is important that the argument connected with a detail is equal to the argument connected with its corresponding unit. For example: `\thm{delen met rest}` corresponds with `\begin{detail}{stpf}{delen met rest}`.

`stpf` stands for *stepwise proof*. Every step or case in a stepwise proof should be started by:

```
\step{#1}
\case{#1}
```

start with	end with	text book:	computer book:
`\begin{detail}{cons}{#1}`	`\end{detail}`	Constructie.	*link*
`\begin{detail}{exam}{#1}`	`\end{detail}`	Voorbeeld.	*link*
`\begin{detail}{exer}{#1}`	`\end{detail}`	Opgave.	*link*
`\begin{detail}{proof}{#1}`	`\end{detail}`	Bewijs.	*link*
`\begin{detail}{stpf}{#1}`	`\end{detail}`	Bewijs.	*link*

Table 4: Kinds of detail

respectively, where #1 states the claim of the step or case. In the text book the proof of this claim, or the treatment of the step will follow immediately. In the computer book all claims will be listed first. Part of this text will be click-able and will lead the user to another window where the proof of a claim will be displayed.

4.6 Computer Algebra parts

The command

```
\begin{matica}{#1}{#2} ... \end{matica}
```

will invoke a link to a Mathematica session with a notebook as argument in the computer book. This link is in the window of the item with name #1. The text between the begin and the end will be converted to a Mathematica notebook that will be living in a file with name #2.ma. In the text book the contents of the notebook will appear.
Similarly,

```
\begin{maple}{#1}{#2} ... \end{maple}
```

could in the future work for Maple in the same way as the above for Mathematica.
The command

```
\begin{code} ... \end{code}
```

tells the text book that it should start to typeset as code. It tells the script `splits` that it will encounter program code.

4.7 Solutions to exercises

```
\begin{sol}{#1} ... \end{sol}
```

Tells the script `splits` that it should write everything between the begin and the end of the LaTeX environment to the solve file. In the future, it will also invoke a link in the computer book in the window that contains the exercise with unit name #1. The argument #1 should be the same for an exercise and its solution, to make communication possible.

`\begin{noprint} ... \end{noprint}`

When LaTeX encounters a `\begin{noprint}` in its source file it starts doing nothing, except looking for an `\end{noprint}`. Therefore, every text that should not appear in the text book, but should appear in the computer book, has to live in between this command. Warning: in general it is not possible to encapsulate this command in the definition of other commands.

4.8 Extra commands

`\history{#1}{#2}`

Creates a link to the History of mathematics pages. #1 should be the name of the mathematician as written on these pages, #2 is the text that will contain the link.

`\link{#1}{#2}`

Creates a link to the URL #1, when in the XHDVI book. Will display text #2 in all books.

`\listofmaticas`

Creates a click-able list of Mathematica Notebooks in the computer book. This command should typically be used in the root file. It is not implemented for the text book yet.

4.9 Markers

Some markers are used. They have the following meaning:

Symbol	Command	Meaning
♣	`\fun`	required
♡	`\votary`	for devotees
♠	`\assiduous`	this is hard work
♢	`\research`	investment required here
¶	`\tickel`	stand alone application available

Table 5: Markers

4.10 Some remarks

1. By editing "textbook.tex" a little bit authors can achieve that `\begin{maple}`, `\begin{matica}` and `\begin{sol}` will have the same effect as `\begin{noprint}`. Just search for the word `switch` in the "textbook.tex" file.

5 Creating a paper version

Run LaTeX three times on the file `root.tex`. Make an index with e.g. `makeindex`, and run LaTeX another time, now inputting the file `root.ind`. Run `dvips` on `root.dvi` and you are done.

6 Creating a XHDVI version

6.1 Preparing the chapter files

The following should be done with every chapter. As an example we will consider a chapter that is living in a file `reken.tex` in a directory algebra. We will work in a directory called `work`

```
> cp ~/algebra/reken.tex ~/work
> cd ~/work
> splits reken.tex
```

6.2 Running LaTeX

Perform the following steps in the directory `work`:

```
> latex root_dvi
> latex root_dvi
## Protect temporary files:
> chmod -w *bl*
> chmod -w *se*
## LaTeX still wants to write on them. As it cannot do this anymore, we'll have
## to give it something else, for example a lot of zets which we have put
## in a file called "z"
> latex root_dvi < z
## Now we can create an index:
> makeindex root_dvi.idx
## But of course we want a nice index with links, "hinde" will do this for us:
> hinde root_dvi.ind
## Now LaTeX the result again after including the file "root_dvi.ind" in
## the root file root_dvi.tex.
> latex root_dvi < z
## Now you can have a look at the result:
> xhdvi root_dvi &
```

7 The setup of XHDVI

7.1 What is XHDVI?

XHDVI is a hacked version of the well-known DVI previewer XDVI. It is the same as XDVI, except that it is aware of hyper-links. These hyper-links are embedded in the TEX source with the help of TEX's `\specials`. So, the special thing about XHDVI is that it is aware of some of these specials. XHDVI is free software and can be obtained via FTP on the HyperTeX site on the World Wide Web, see [3]. XHDVI is currently only available for most of the UNIX platforms. One will need a low level macro that will make these specials available for the user. We used the macro `hyperbasics.tex`. A description of the commands defined by this macro can be found in section 14.

7.2 Installing XHDVI

How to install the source or binaries is also described on [3]. The difficult thing about XHDVI is to let it work with links to local non-DVI files and to any file on the WWW. The XHDVI program should handle links to local files by itself. Unfortunately it doesn't: its sends the string describing the link to its WWWBROWSER. One can choose this program by setting the environmental variable `WWWBROWSER`. We made our own WWWBROWSER; it is called `openit`. So one should put something like

```
> setenv WWWBROWSER $HOME/bin/openit
```

in one's initialisation file, e.g. `~acela\.login` (depending of course on your local settings).

7.3 Openit

The program `Openit` takes a string as its input. If the string starts with `http://` it is obviously a URL so it starts a real WWW browser (like `Netscape` or `Mosaic`) to handle the link. If the string starts with `file://` it looks like a local file so it will try to do something with that. In all other cases it will return an error message.
In the case of a local file it will first determine the extension. It will look for this extension in a file called `.mime.types` that it expects to find in the home directory of the user. MIME stands for Multipurpose Internet Mail Extensions and is set up by the World Wide Web people to specify file types in a straightforward way. Suppose now the string that is given to `openit` looks like

```
file://ufs1/riaca/lindenb/cas/matica/quot.ma
```

The extension in this case is `.ma`. My version of the file `.mime.types` contains a line

```
application/xmathematica        ma
```

so `ma` is bound to the application `xmathematica`. Now we have to find the actual program and this information is present in a file called `.mailcap`, that should also live in your home directory:

```
application/xmathematica; /usr/local/bin/mathematica %s
```

So, now we can start up our local file because `mathematica` accepts an argument (`%s`); we can substitute our filename in that place. However, not all programs accept an argument, `MapleVr3` for example does not.

8 Creating a PDF version

8.1 Introduction

The first part of the process is similar to the creating of the XHDVI version. The root file is now called `root_win` instead of `root_dvi`. Two reasons make it necessary to use another root file now. Firstly, the dvi file will serve as input for a special version of the dvips converter that conserves most hyper-links, but will crash on all non-internal links, so these links have to be removed with help of an adapted macro. Secondly, the resulting postscript file has to be converted into a PDF file that will live on a PC running Windows. This PC does not know about the default fonts used by postscript, so we have to do something special here too.

8.2 `dvihps`

`Dvihps` is a hacked version of the well known UNIX program `dvips`. It is the same except a new option `-z` is added that will conserve all internal links that are available as specials in the dvi file. Information about obtaining and installing the program is again available via the HyperTeX site, see [3].
Note that `dvihps` will stop as soon as it finds a link to an anchor that is not present, so make sure that your links and anchors are all correct before you start converting the dvi file.

8.3 Lucida fonts

Lucida fonts are *scalable* fonts. That means that fonts are not defined in a restricted number of fixed sizes, but that only the outlines of the fonts are defined, and that these outlines can be scaled to every size that is requested. Lucida fonts are not free. They can be obtained from *Y&Y inc.*, see [2]. One can tell TEX to use this fonts by inputting a macro called `lcdlatex.tex`. When everything is installed properly this is almost the only thing you have to worry about. (The *almost* will be explained in subsection 8.5.) More information on Lucida fonts is available via the Y&Y–homepage, see [2]. Please note that it is only permitted to distribute these fonts on special terms. A flyer called `Acrobat.txt` can be obtained from Y&Y that lists these conditions.

8.4 Portable Document Format

A few years ago, Adobe introduced the Portable Document Format. It is a format that supports high quality graphics and typesetting. Adobe developed it, because they wanted to have a format that is platform independent and good looking. One can view PDF documents via the Adobe Acrobat Reader that is available free of charge from the Adobe Acrobat homepage, see [1]. Probably it will be possible to look at PDF files with Netscape as well in the not too distant future. (In fact it is already possible in Windows NT and Windows '95, combining Acrobat Amber and Netscape, see [4]. One can down-load examples of PDF files via the Adobe Acrobat WWW page as well. We are using the Adobe Acrobat Pro software kid. It consists of:

Adobe Acrobat Reader A PDF viewer.

Adobe Acrobat Distiller A program that converts postscript to PDF.

Adobe Acrobat Exchange A program that enables the user to manipulate PDF files. Some features are:

- adding navigational links.
- annotating the file.
- cropping pages (e.g. for removing superfluous white space).

Some other programs we didn't use.

When we bought this kid, it was not available for UNIX at all. By now SunOS(TM) and Sun(TM) Solaris are supported as well by Adobe Acrobat.

8.5 Ligature problems

Before we are able to leave UNIX and continue on PC, one problem has to be solved. TEX uses ligatures. This means that certain combinations of characters are joined in one sign. This happens for example with *fl*, *fi* and *ff*. This works fine when one sticks to UNIX, but unfortunately the PC does not understand this ligature feature and funny characters will appear in the final PDF document. To avoid this problem one has to achieve that these combinations never occur. This can be done by replacing all occurrences of these patterns by e.g. *f{}l*, *f{}i* and *f{}f*. The program `liga` takes care of this. So what one has to do to create a PDF file:

8.6 TEX to PDF

```
> cp ../algebra/reken.tex .
## solve the ligature problem:
> liga reken.tex
> splits reken.tex
> latex root_win
    ...
> latex root_win < z
> cd ~acela/dvihps_0.3c
> cp ~acela/rod/voor/root_win.dvi .
## converting to hyperpostscript:
> dvips -z root_win -o root_win.hps
## now move the file to the PC.
##  convert the file to PDF by using Adobe Acrobat Distiller
> c:\acrodist\acrodist root_win.hps
## Add extra links, crop pages, etc. in Adobe Acrobat Exchange
> c:\acroexch\acroexch root_win.pdf
## Look at the result with the free reader:
> c:\acroread\acroread root_win.pdf
```

9 The setup of Acrobat Reader

Once the pdf file is ready one should think of distributing it and install it in such a way that all interactive features really can be performed. To achieve this the following should be done:

1. Create a directory `c:/algebra/` (for example) with subdirectories `pdf`, `matica` and `tcl`.
2. Copy the pdf file into the newly created `pdf` directory; copy the Mathematica notebooks into the `matica` directory and the Tcl applications into the directory `tcl`.
3. Associate the `.tcl` extension with the `wish.exe` application. (That is the main program created during the installation of Tcl/Tk on Windows.) Also associate `.ma` files to Mathematica or the Mathematica Reader. To associate: go to File Manager and choose from the Menu File the option associate. Enter an extension (e.g. `.ma`) and use Browse to find the right application (e.g. `C:\MATHREAD\MATHREAD.EXE`). When afterwards in File Manager you will click on a file with the extension `.ma` the application now associated with it should appear, running the file you did press.
4. Use for example the programs `Winzip` and `Osetup` to put these files on floppy, conserving the directory structure and file associations.

10 The Mathematica Notebooks

The source document contains, among others, special Mathematica units, marked by a `{detail}{matica}` tag. These Mathematica units consist of pieces of text and pieces of code. The code is in between tags `\begin{code}` and `\end{code}`. The program `splits` will write every Mathematica unit to its own file; while doing this it searches for `code` tags. This behavior enables `splits` to make automatic the production of Mathematica notebooks: every time a change from text to code or the other way round occurs, a new Mathematica `subsubsection` will be opened and the old one closed.
These subsubsections will be subgroups from the main group that bears the title of the unit. As the code was in verbatim mode in the source document, one doesn't need to edit this anymore. The text contains LaTeX commands and pieces in LaTeX's math mode. When some commands occur a lot one could change `splits` in such a way that these commands are handled correctly. Otherwise it might be faster just to edit the primitive notebook that `splits` delivers.

11 The Tcl/Tk applications

11.1 The Tcl/Tk language

Tcl stands for Tool command language and it is a scripting programming language: one doesn't need to compile a program, but it will be interpreted by binaries called `Wish` or `Tclsh`. Tk stands for Toolkit: it presents a library of graphical user interface elements that one can use with Tcl. Tcl/Tk is obtainable free of charge from [5]. Beta releases are available for Macintosh and PC's running Windows. A stable version is available for most UNIX platforms. The inventor is called John Ousterhout and he did also wrote a clear manual for the old version, see [10]. Brent Welch wrote a book that covers Tcl 7.4 and Tk 4.0, see [11]. Both books are available in draft version on the Internet.

11.2 MultTable

`MultTable` is an application meant for students that start learning about groups. The student is asked to create a multiplication table for a group of order not passing 12. When the user indicates so, errors will be reported by the program. When Tcl/Tk is installed on your local system you can down-load it from the `MultTable` home page, see [6].

11.3 RSA

`RSA` is an application based on the famous RSA crypto system. The user is asked to fill in some parameters. When a mistake is made, the program will indicate this and tries to help. When all parameters are correct, the user can encode a number or a text, depending on the version.

12 Known problems / To Do

12.1 Computer Algebra Systems

Maple worksheets have not been created yet at all. The program `splits` cannot do clever things with worksheets yet.

12.2 In the TEX macro's

Temporary files Table of contents like pages are created at the beginning of each chapter and at the beginning of each section. In order to do this, temporary files are produced during the TEX compilation. TEX only allows about sixteen writeouts (c.q. open write connection to a file) at a time. We need much more temporary files. We tried to work around this by closing temporary files when they were done with, but didn't succeed completely: the temporary files have to be write protected after the second LATEX run.

Cleaning the macros Subject specific notation commands like `\spoor` and `\kgv` are in the same macro as the new commands, described in 4. These subject specific notation commands should live in their own file or in the same file as the hyphenation patterns (that is `hyphalg.tex`).

Tables of contents When using the macro file `hyperlatex.tex` the only thing that is clickable in a table of contents like page is the page-number. It would be much nicer if one could click on the name of a chapter to go to this chapter. This is achieved within the command `\listofmaticas`. The command `\listofmaticas` works fine for the book that runs on UNIX.

Hyperref.sty In the book that runs on PC two problems arose:

1. The program `dvihps` cannot handle the links to external files (the Mathematica notebooks in this case) in the way they were defined by the macro `hyperlatex.tex`.
2. Some areas that catches the mouse click are too small.

Probably one can get rid of both problems by using the macro file `hyperref.sty` by Sebastian Rahtz, see [3]. As this style file is developed for LATEX2e and our macro's are developed for LaTeX209 one could do two things: firstly, updating our setup to LaTeX2e; secondly, converting `hyperref.sty` to LaTeX2e. The second option seems the most attractive: it would be easier to get support from other developers afterwards and it would be a step ahead instead of a step back. It seems that the macro file `hyperlatex.tex` is now no longer supported.

Language problems Some commands described in Section 4 produce words. For example: `\begin{theorem}..` will cause LaTeX to put the word **Stelling** in the dvi file. When the source document is e.g. French, this word is of course unwanted. At this moment one would have to search for all occurrences of the word **Stelling** in three macro files and replace it by **Theorème** In general all occurrences of language specific words somewhere deep in the macros should be replaced by more general tags like `\thmname`. One could think of:

1. Following the Babel scheme, described in [8].
2. Creating an easy to edit piece of macro that contains all language specific definitions for together with an **if .. then ..** structure. (This is of course exactly what Babel does).

12.3 In the Tcl/Tk applications

Tcl/Tk for Windows The two described Tcl applications work fine on Unix. On Windows however the MultTable applications stops testing after a while and the RSA program looses some buttons during encoding of a text. These are probably bugs in the Tcl/Tk distribution for Windows. This distribution was obtained via [5].

12.4 In the general setup

Make file A make file should become available for the programs that run on UNIX in order to avoid as much installation problems as possible for future authors; in fact one should think of two make files: one for readers and one for writers.

Create script A create script should become available for both the creating of the UNIX book and the creating of the Windows book, as far as this process runs on UNIX.

Setup.exe For the distribution of the Windows book it would be nice if a user should have to run only one file called `setup.exe` that would install everything needed on user's PC, prompting user for information when necessary.

13 Ideas

13.1 In the Algebra source document

Using normal LaTeX commands It *may* be a good idea to use standard LaTeX commands as much as possible. The command `\startsection` could be changed in `section`; `units` are on the same level as LaTeX's `subsections`. On the other hand: this may become very confusing.

13.2 Computer Algebra Systems

Library files For every CAS for every chapter, a library file should be available that contains all procedures of that chapter. One could think of having a link to a open CAS section that has this library file included however. Students can use this scratch Notebook to practice with the procedures.

New kernel every time When a link to a CAS is followed two times, the complete interface will be started two times also. It would be much better if the user was prompted if the old interface should be used again or not. In general, this cannot be changed by us but only by the developers of the specific CAS. This feature is built in in the Netscape navigator and is called **Remote control**.

Automizing the creation of Notebooks When more structure would be available in the source document (like *matica–sections* corresponding with Notebook–subgroups) nicer looking Notebooks could be created in an automized way. The same holds if the program `splits` would be aware of some basic LaTeX commands like `bf`, `tt` and `it`.

13.3 Prerequisites

Something like `\begin{prereq} ... \end{prereq}` is not implemented yet. Probably one does not want to see prerequisites in the text book. In the computer book, links to prereq pages could appear in units that contains a new definition or a new theorem. A prereq page could contain the following items:

1. General prereqs, like "It is a well known fact in Linear Algebra that ..."
2. Links to a "definition tree".
3. Links to a "theorem tree".
4. Links to other, directly available, documents, on the Internet or somewhere on the local file system.
5. References.

A definition tree should be a click-able graph. On top is the current definition. Vertices correspond with definitions. A edge is going down from one definition to a second definition if the second definition is needed to define the first one. The number of definitions on one page should be restricted to avoid that the screen will turn black at the end of the document. The same kind of construction can be used for the theorem tree.

13.4 Navigation

Subway map It would be nice if on the left side of the screen in the computer book some place is reserved for a kind of map that helps the user to find its way. This map should be click-able and give he user the opportunity to jump easily to another unit or another chapter. One can find examples in the Adobe Acrobat Software.

Clickable header Another feature that could help the user to locate him/herself in the book is to add a header that displays the current chapter, the current section and the current unit, or whatever is appropriate. These headers should be click-able as well.

13.5 Stand alone applications

Two bigger and one smaller stand alone application have been created so far. Some other ideas that arose during the year are listed below:

Visualizing simple algorithms Simple algorithms can be visualized by a little program that performs the steps in the algorithm slowly, so that the user can see what happens. Candidates could be the Sieve of Eratosthenes, the Bubble Sort algorithm (I have seen a Java Applet that compares different sorting applets), Orbit algorithm etc.

Detection pattern Let the user enter a p and a g. Give a picture of the detection pattern as a result.

Polynomial Calculator I started on it already. Enter some polynomials, choose a ring, perform a operation like multiplying, gcd, factorization.

Coding Theory

Permutations A simple version of Lloyd's game is available. It needs a good random (Usage of the clock) and some questions attached to it. It would be nice if a user could also enter its own starting configuration and if a cheat button could tell the user if it is possible to finish the game at all.

Graph program Draw a plain graph interactively, adding vertices and edges with the mouse. Compute the automorphism group afterwards.

Monoids The Multiplication Table program can be extended in many ways:

1. It doesn't support Monoids yet.
2. It is not able to finish a multiplication table when it could do it mathematically spoken.
3. it is not connected to Computer Algebra Systems. One could create a small program creates a group in GAP and draws a picture of the multiplication table in MultTable afterwards.
4. ...

14 The commands defined by hyperbasics.tex

This section presents an overview of the kind of links one can create using the macro-file hyperbasics.tex, written by Tanmoy Bhattacharya. Links are created automatically by the macro-file hyperlatex.tex. These macros can be used by inputting them in your (La)TeX document in the preamble. So it should look like this (in for example old LaTeX):

```
\input /ufs/lindenb/macros/hyperbasics.tex

\documentstyle[12pt]{report}
```

14.1 Links inside a dvi document

Like in HTML it is possible to set anchors in a text. When one wants to jump to a specific place in a document, this specific place should be an anchor. Anchors can be referred to in two different ways: **1.** all anchors have a name. When this name is unique one can use this to refer to when putting a link; **2.** every anchor lives in a category. In this category an anchor is made unique by its category name. For example: in the anchor

```
\hyperdef\myname{page}{1}{}{any text you like}
```

the name of the anchor is "myname", it is living in the category "page" and its unique category name is in this case "1". So one can make a link to this anchor by

```
\hyperref{}{page}{1}{Go to page 1 or some other text}
```

using its category name and its category, or by:

```
\hyperref\myname{Go to page 1}
```

using the name of the anchor.

14.2 Links inside a server

One can link to another dvi-file (in this case called otherfile.dvi) like this:

```
\hyperref{file:otherfile.dvi}{example}{7}{Go to
                         the example in the other file}
```

or, when it is OK to start with the first page by

```
\hyperref{file:otherfile.dvi}{}{}{Go to the other file}
```

or, equivalently

```
\href{file:otherfile.dvi}{Go to the other file}
```

leaving the anchors for what they are. It is not possible in this setting to use something like `\hyperref{file:otherfile}\myname{to myname}`

In fact you can put here every kind of file, where the kind is specified by the extension of the file. XHDVI is then supposed to look up the extension in the .mailcap and .mime.types files (or .mime-types). When this doesn't seem to work (as is the case with me), you can let another program (like Netscape) do the job. You can specifying this program by the environmental variable `WWWBROWSER`.

Links to arbitrary local files look like this:

```
\hyperref{file:/ufs1/riaca/lindenb/demo/
         matica/delenmetrest.ma}{}{}{Mathematica notebook}
```

or, equivalently

```
\href{file:/ufs1/riaca/lindenb/demo/
         matica/delenmetrest.ma}{Mathematica notebook}
```

Something funny like this also works in a way:

```
\special{html:<img src="/ufs1/riaca
                      /lindenb/postscript/platypus.eps"}
```

It starts up ghostview at the same time as XHDVI, displaying the image ”platypus.eps” but it doesn't create anything in the dvi-file.

14.3 Links on the World Wide Web

Again, the variable WWWBROWSER should be set properly. After this is done, following a link like

```
\href{http://xxx.lanl.gov:80/hypertex}{To hypertex homepage}
```

or, equivalently

```
\hyperref{http://xxx.lanl.gov:80/hypertex}{}{}{To hypertex homepage}
```

should start up your World Wide Web browser, and make it display the Hypertex homepage.

15 Overview of software tools used

name	description	directory	remarks	author
hyperbasics.tex	defines hyperlinks	acela/tex		outside
hyperlatex.tex	creates automatic links	acela/tex		outside
textbook.tex	text version macro	acela/tex		RCL; Van Leeuwen
hypbook.tex	XHDVI/Unix macro	acela/tex	used by the main level	RCL
winhyp.tex	Win3.1/Acroread macro	acela/tex	used by the main level	RCL
hypdet.tex	computer version macro	acela/tex	used by the details	RCL
splits	splits up a chapter	acela/bin	Tcl program	RCL
XHDVI	displays hyper-linked dvi	canc: /usr/local/bin		outside
openit	handles XHDVI links	acela/al../lib/i../tcl/	TCL	RCL; Dickie
hinde	creates hyperindex	acela/bin	TCL program	RCL
liga	Win3.1/Acroread	acela/bin	Ligature program	RCL
.mailcap	handles XHDVI links	acela	text	WWW
.mime.types	handles XHDVI links	acela	text	WWW
acroread.exe	displays pdf	C:/acroread/	Icon + assoc.	Acrobat
acroexch.exe	*edits* pdf	C:/acroexch/	Icon	Acrobat
acrodist.exe	postscript2pdf	C:/acrodist/	Icon + assoc.	Acrobat

Table 6: Software tools used

References

[1] http://www.adobe.com/Acrobat/ – Adobe Acrobat home-page, (1996).

[2] http://www.YandY.com/Welcome.html – Home-page of Y&Y, Inc., Programs and fonts for the TeX typesetting system, (1996).

[3] http://xxx.lanl.gov/hypertex – Home-page of the Hypertex project, (1995).

[4] http://www.netscape.com/ndx.html – Netscape home-page, (1996).

[5] http://www.sunlabs.com/research/tcl/ – Home-page of the Tcl project, (1996).

[6] http://www.can.nl/~lindenb/MT/mt.html – MultTable homepage, R.C. Lindenbergh, (1995).

[7] Cohen, A.M., Cuypers, H., Sterk H. – Algebra, lecture notes, Technische Universiteit Eindhoven, (1995).

[8] Goossens, M., Mittelbach, F., Samarin, A. – A LaTeX Companion, Addison-Wesley, Reading et al., (1994).

[9] Knuth, D.E. – The TeXbook, Addison-Wesley, Reading et al., (1984).

[10] Ousterhout, J.K. – Tcl and the Tk Toolkit, Addison-Wesley, Reading et al., (1993).

[11] Welch, B. – Practical Programming in Tcl and Tk, Prentice Hall, (1995).