Tampereen teknillinen korkeakoulu
Julkaisuja 183

Tampere University of Technology
Publications 183

Jari Multisilta

Hypermedia Learning Environment for Mathematics

Tampere 1996

Tampereen teknillinen korkeakoulu
Julkaisuja 183

Tampere University of Technology
Publications 183

Jari Multisilta

Hypermedia Learning Environment for Mathematics

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Auditorium NA032, at Tampere University of Technology, on the 14th of June 1996, at 12 o'clock noon.

Tampere 1996

ISBN 951-722-561-X (printed)
ISBN 952-15-1505-8 (PDF)
ISSN 0356-4940

Abstract

This thesis describes the design, implementation and use of a software package for hypermedia based learning on mathematical sciences. The software package integrates hypertext, computer-aided exercises, graphics, videos and sound into a hypermedia based learning environment (HMLE).

The design and implementation problems of hypermedia based learning environments in mathematics are presentation, input, comparison and evaluation of mathematical expressions, document conversion from ordinary text to hypertext, division of linear text to hypertext and automatic link generation. As a solution to these problems, HMLE consists of document conversion tools for Microsoft Word documents, presentation tools for mathematical hypertext documents and authoring tools for automatic link generation.

In order to support learning, problem-solving tools and other cognitive tools should be integrated to hypermedia environment. As a solution, HMLE introduces interactive exercises generated by Mathematica, exercises with hints, links to animation and movie applications, links to mathematical applications, and concept maps.

Finally, classroom experiences of hypermedia based learning environments are presented.

Preface

This work was done at the Department of Mathematics, Tampere University of Technology, during the period of 1989-1995. I want to express my gratitude to the head of the Hypermedia Laboratory, ass. prof. Seppo Pohjolainen, for his support. Also, I want to thank M. Sc. Kostadin Antchev, M. Sc. Markku Luhtalahti and M. Sc. Kari Suomela for cooperation and helpful discussions. I am also grateful to Lis. Fil. Pentti Hietala for his valuable comments for improving the introductory part of this thesis. For the revision of the English manuscript I am grateful to Mr. James Rowland and Ph. D. Robert Piché.

This work was supported by the Finnish Academy, the Finnish Ministry of Education and Tampere University of Technology. They are gratefully acknowledged. Also, I would like to express my gratitude to the Jenny and Antti Wihuri Foundation, the Pirkanmaa Cultural Foundation and to the Ulla Tuominen Foundation for their support in the form of personal grants.

I would like to express my gratitude to my wife Teija, and my children Jenni and Hanna for giving me the chance to live a full life as a husband and father while I was writing my Ph. D. thesis.

Finally, I would like to dedicate this work to my father Ossi who always supported me in my studies in his own way. Unfortunately I can not share the joy of completing the thesis with him, as he was called to eternity on the eight of August, 1995.

Publications

Paper 1. Multisilta J., Pohjolainen S.: Hypermedia and Animation in Distributed Parameter Systems Education. *International Journal on Mathematial Education in Science and Technology.* 26(4): 599-618, 1995.

Paper 2. Pohjolainen S., Multisilta J., Antchev K.: Hypermedia Learning Environment for Mathematical Sciences. In Kajler N. (ed.): *Human Interaction in Symbolic Computation*; Texts and Monographs in Symbolic Computation, Springer-Verlag. To appear.

Paper 3. Antchev K., Multisilta J., Pohjolainen S.: Interactive Exercises for Matrix Algebra. Submitted to *Mathematica Journal*.

Paper 4. Multisilta J., Antchev K., Pohjolainen S.: Hypermedia for Mathematics: Authoring Courses with HMLE. *Proceedings of World Conference on Computers in Education*, WCCE'95 conference, Chapman & Hall, July, 1995.

Paper 5. Antchev K., Luhtalahti M., Multisilta J., Pohjolainen S., Suomela K.: A WWW Learning Environment for Mathematics. *World Wide Web Journal*, Issue One: Conference Preceedings, Fourth International World Wide Web Conference, Boston, Massachusetts, 11.-14.12.1995.

Paper 6. Pohjolainen S., Multisilta J., Antchev K.: Matrix Algebra with Hypermedia. Submitted to *Education and Information Technologies*, IFIP TC 3 Journal.

Abbreviations

ACM	The Association for Computing Machinery
ASCII	American Standard Code for Information Interchange
API	Application Programming Interface
CAI	Computer Assisted Instruction
CAL	Computer Assisted Learning
CAN	Computer Algebra Netherlands
CAS	Computer algebra system
CBT	Computer Based Training
CD	Compact Disk
CD-R	Compact Disk Recordable
CD-ROM	Compact Disk Read Only Memory
CNVR	Converted RTF files. A hypertext document format for HMLE
COSTOC	Computer Supported Teaching Of Computer Science
CTI	Computers in Teaching Initiative
CTICMS	CTI Centre for Mathematics and Statics
DISM	Distributed mathematical information system
DLL	Dynamic Link Libraries
DTD	Document Type Definition
DVI	Device Indipendent Form
GLSS	General Lesson Specification System
ETO	Essential Tools and Objects
GIF	Graphics Interchange Format
GUI	Graphical User Interface

HMLE	Hypermedia based Learning Environment
HTF	Hypertext format for Hyper-G documents
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ISO-9660	International standard for CD-ROM
ISO-8879	International standard for structured documents (SGML)
JPEG	Joint Photographic Experts Group
MPEG	Motion Picture coding Experts Group
OLE	Object Linking and Embedding
OOP	Object Oriented Programming
PDF	Portable Document Format
PLATO	Programmed Learning and Teaching Operation
QT	QuickTime digital movie and animation format
RIACA	Research Institute for the Applications of Computer Algebra
RTF	Rich Text Format
SGML	Standard Generalized Markup Language
TCL	Think Class Library
TE	TextEdit Manager in Macintosh system toolbox
TLTP	Teaching and Learning Technology Programme
TMP	Transitional Mathematics Project
TUT	Tampere University of Technology
URL	Universal Resource Locator
WAV	Microsoft Windows Audio File Format
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XCMD	External Command for HyperCard
XFCN	External Function for HyperCard

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computers have played an important role in education since the beginning of the computing era. For a long time, computers and educational software were seen as a replacement for teachers and it was believed that computers can reduce the costs of education. Quite often these expectations have failed. Even so, computers can support learning by providing a learning environment that is not possible to build under ordinary circumstances. For example, experiments with critical systems can be done by using computer simulations (chemical simulations, flight simulators etc). Computers can also support learning by providing new ways to access information, for example by using hypermedia.

In many application areas it is possible to create real-world problems with computers by using hypertext and hypermedia. To solve these problems using the computer can be considered as a modern learning environment.

1.1 Research problems

The idea of a hypermedia based learning environment for mathematical sciences originated in 1989 when the author started to implement a distributed parameter systems simulator for educational purposes together with professor Seppo Pohjolainen. Positive response from the simulator encouraged us to design a general, hypermedia based learning environment for mathematical sciences (Paper 1).

Problems of mathematical CAL software are studied for example in [15]. In

general, the problems are presentation, input, comparison and the evaluation of mathematical expressions. Problems in producing hypermedia material are also well-known: document conversion from ordinary text to hypertext, division of linear text to hypertext nodes and automatic link generation [13], [14], [26], [45], [91]. In order to support learning, problem-solving tools (like mathematical software packages) and other cognitive tools should be integrated to hypermedia environment [16], [52], [61], [66], [92], [93]. Finally, distribution of hypermedia material can be considered a problem because hypermedia based learning material should be easily available and it should be cheap to distribute.

In this thesis, these problems are analysed and solutions are suggested in order to implement a hypermedia based learning environment for mathematical sciences, namely HMLE. The requirements for hypermedia based learning environment for mathematical sciences are presented in Papers 1 and 2. In general, the requirements are to:

A create a database of mathematical hypertext, computer-aided exercises, graphics, videos and sound.

B support self-study by having all the material prepared for personal computers and distributed on CD-ROMs.

C minimize the cost, the plain hypertext should be readable without the need for any commercial software.

D support problem solving, commercial software should be easily integratable into the learning environment when available.

E make authoring, maintenance and updating of the material as simple as possible using automatic link generation and implicit linking.

F record to some extent the actions of the student in order to examine different study styles.

In table 1.1 the research problems and requirements for hypermedia based learning environments are summarized.

	Research problem	Corresponding requirement(s)
RP 1.	Presentation of mathematical texts.	A,C
RP 2.	Input, comparison and evaluation of mathematical expressions.	A,D
RP 3.	Conversion from ordinary text to hypertext.	E
RP 4.	Division of linear text to hypertext nodes.	A
RP 5.	Automatic link generation.	E
RP 6.	Integration of problem-solving tools and other cognitive tools into hypermedia environment.	D,F
RP 7.	Distribution of hypermedia material.	B

Table 1.1: Summary of research problems

The main achievement is not only separate solutions to the research problems mentioned above, but the learning environment that combines hypermedia, authoring tools and learning tools to hypermedia based learning environment for mathematical sciences (i.e. sciences that contain structured mathematical information, mathematics, physics or control engineering).

1.2 Structure of the thesis

Educational technology is a discipline whose main object is to study the relation of education and technology [90]. It includes instructional design and planning, teaching methods (e.g. simulations), instructional media (such as computers and hypermedia), instructional resources, learning (including study skills, learning theories, motivation and problem solving), and assessment and evaluation [58].

The main contribution of this thesis belongs to educational technology. However, the design and implementation of the hypermedia based learning environment is greatly influenced by the subject matter (mathematics).

This thesis concentrates on the design and implementation of hypermedia based learning environment and the authoring tools. Althought the point of view is tech-

nologically oriented the design is based on the modern learning theories presented in subchapter 1.4. The software packages presented in this thesis have been used in a classroom and student feedback is discussed in Chapter 6. However, the study of the question "Do students learn better using hypermedia than using books and going lectures" is beyond this thesis.

A theoretical framework and concepts are presented in subhapter 1.4. In Chapter 2 a short history of CAL and hypermedia is outlined. There is also a review of related work and applications in this area. Chapter 3 concentrates on markup languages used in hypermedia documents and communication protocols used in communication with mathematical applications. Chapter 4 discusses the conversion problem and presents a solution to the conversion of mathematical documents to hypermedia. Chapter 5 discusses the implementation issues of HMLE and presents a solution to the presentation problem of mathematical equations. Finally, Chapter 6 discusses classroom experiences of HMLE.

Paper 1 discusses distributed parameter systems simulator for educational purposes that can be considered as a first step towards a hypermedia based learning environment. It is related to research problem RP 6. Paper 2 discusses the design and architecture of HMLE and presents a few scenarios of how HMLE can be used in learning mathematics. In Paper 2 the Exercise Maker, assessment software for HMLE is also presented. Paper 2 is related to problems RP 1, RP 4 and RP 6. Paper 3 concentrates on the implementation of Exercise Maker and authoring exercises for Exercise Maker and is related to problems RP 2 and RP 6. Finally, Paper 4 discusses the authoring of hypermedia documents in HMLE and is related to problems RP 3, RP 4, RP 5 and RP 6. Paper 5 discusses the components and implementation of hypermedia based learning environments on the World Wide Web (WWW) and is related to research problems RP 1, RP 3, RP 5, RP 6 and RP 7. Paper 6 discusses the Matrix Algebra hypermedia course, the role of video clips in HMLE and different study styles in hypermedia based learning environments and is related to reseach problems RP 2 and RP 6.

The thesis should not be read linearly but more like hypertext. In general, the first part of the thesis (Chapters 1-7) presents a theoretical framework and historical

overview of the topic. In addition, there is background information (Chapters 3 and 4) for understanding the technical description of the implementation of the software. The second part (the Papers) concentrates more on restricted topics such as simulation and animation as part of a hypermedia or authoring for mathematical hypermedia. The suggested reading order is Chapters 1 and 2, Paper 1, Chapters 3 and 4, Paper 2 and 3, Chapter 5, Paper 4 and Chapter 6, Papers 5 and 6, and Chapter 7.

1.3 Author's contribution to the papers

For Paper 1 the author designed the user interface for TDP simulator together with Seppo Pohjolainen. The author then implemented the user interface and ported TWODEPEP simulation software from VAX to Macintosh. Finally, the author designed and implemented the animation software for the solutions of TDP simulations and performed the simulation examples presented in the paper. The TDP software was tested in the classroom by Seppo Pohjolainen and the author.

For Paper 2 the author designed the architecture of HMLE software. HMLE was also implemented by the author except Exercise Maker, that was designed together with Kostadin Antchev and Seppo Pohjolainen. Finally, the author has used the software in the classroom together with Seppo Pohjolainen.

For Paper 3 the author designed the Exercise Maker together with Kostadin Antchev and Seppo Pohjolainen. Exercise Maker was implemented by Kostadin Antchev. The author has used the software in the classroom and designed exercises for Exercise Maker.

For Paper 4 the author designed method for authoring HMLE courses i.e. the conversion of linear mathematical texts to hypermedia, concept maps and interactive theoretical exercises.

For Paper 5 the author designed the structure of hypermedia based learning environments on the WWW and converted the Matrix Algebra lecture notes to the WWW. The conversion of LaTeX documents to the WWW were done by Kari Suomela. The interface to the Mathematica and interactive exercises were done

by Kostadin Antchev. The implicit links in WWW were designed by the author together with Markku Luhtalahti. Markku Luhtalahti did the implementation of the Linktool for WWW.

For Paper 6 the author designed the user interface of Exercise Maker together with Seppo Pohjolainen and Kostadin Antchev. The author did the classroom experiences of HMLE in Matrix Algebra.

1.4 Theoretical framework

In this section the theoretical framework and definitions of concepts are presented. There is no single theory for hypermedia based learning environments. In order to implement a hypermedia based learning environment that supports learning it is necessary to understand hypermedia, design of hypermedia applications and computer aided learning.

1.4.1 Hypertext and hypermedia

Hypertext can be defined as a database in which information (text) has been organised nonsequentially or nonlinearly [29]. The database consists of nodes and links between nodes. *A node* is a block of information presented to the user. In hypermedia applications nodes are sometimes called cards (HyperCard), documents (Intermedia) or pages (World Wide Web).

There can be many links from one node to other nodes, and a node can be referenced by many links. *A link* is defined by source and destination nodes, and by an anchor in the source and the destination node [84]. The *anchor* locates the link on the screen. For example, a rectangle expressed in window coordinates can be an anchor in a picture or a range of characters can be an anchor for a text file. Sometimes anchors are visually emphasized for example by underlining the character range.

The destination of a link can be a file (so-called string-to-lexia link) or a string in a file (string-to-string link) [96]. With a string-to-lexia link it is not possible to reference to a certain part of a file. This kind of link can make hypermedia

easily navigable, especially if the destination nodes are "short" documents. String-to-string links would permit the destination to be a string in a file, but this kind of link requires more planning in the design process. String-to-lexia links also support *implicit linking*. Implicit links are generated by the hypermedia software at runtime, for example referential links from a concept to the definition of the concept. Implicit links are sometimes called *computed links* [84]. In contrast to implicit links, *explicit links* are generated by the hypermedia author.

In some systems the link information is saved to the document itself (World Wide Web). This requires a special file format or the use of document mark-up languages. In other systems the link information is saved separately from the documents (Microcosm). The nodes and links form a network structure in the database. *Hypermedia* is a database which contains pictures, digitised videos, sound and animations in addition to text.

1.4.2 Designing hypermedia applications

There has been a lot of effort to make a general standard for characteristics and functionality of hypertext systems. The Dexter Hypertext Reference model seems to have gained global acceptance as such a model. For example, the Communications of the ACM devoted its February 1994 issue to the Dexter Hypertext Reference Model. Another widely used general hypertext model is the Hypertext Abstract Machine (HAM) presented in [25].

The purpose of the Dexter Hypertext Reference Model is to describe standard hypertext terminology coupled with a formal model of the important abstractions commonly found in a wide range of hypertext systems [50], [51]. The Dexter Hypertext Reference Model is intended to be used as a theoretical framework to which existing hypermedia systems can be compared. Because the architecture of hypermedia based learning environment presented in this thesis consists of many features presented in the Dexter Model, it is discussed here briefly. In Chapter 5 HMLE is compared to the Dexter Model.

The Dexter Hypertext Reference Model was written in the formal specification

language Z. The Dexter Model divides a hypertext system into three layers, the *run-time* layer, the *storage* layer and the *within-component* layer [50]. The layers of the Dexter Model can be seen in figure 1.1.

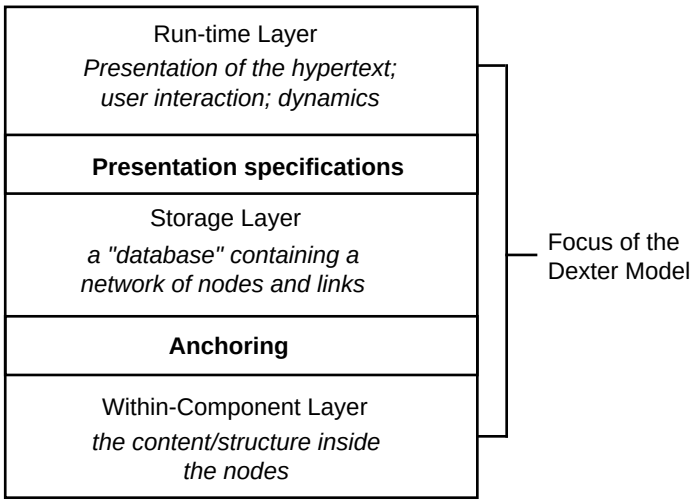

Figure 1.1: Dexter Model.

The storage layer describes a database that contains a network of *components*. A component can be an atom, a link or a composite entity made from other components. Atomic components are used as a synonym for a node in general hypertext terminology. Composite components contain the blocks of text, graphics, images, animations or other components. Link components are a sequence of two or more "end-point specifications", each of which refers to (a part of) a component in the hypertext [50]. Each component in a hypertext has a unique ID number, so called UID.

There are two important functions in the storage layer: *resolver* function and *accessor* function [50]. Accessor function is the mapping from UIDs to the actual components. Resolver function resolves the component UID from the link specification.

Within-component layer describes the contents and structure of a component. In the Dexter Model within-component structure is considered to be outside the hypertext system. Instead, another model designed to model the structure of datatypes inside components should be used with the Dexter Model [50]. The within-component layer provides the anchoring mechanism to the hypertext network [50].

The run-time layer provides a generic model for presenting the hypertext to the user, and provides the tools to user to interact with the hypertext system [50]. The run-time layer uses so called presentation specifications in order to decide how to show a specific part of a component to the user [50]. For example, when a student follows a link to an exercise it is presented as a static text and a dialog for entering the answer. If the teacher follows the same link the system could present the exercise as an editable text.

1.4.3 Computer aided learning and learning environments

There are many concepts that describe software, hardware and learning theories used for teaching and learning with computers. Depending on what aspects of teaching and learning are emphasized different concepts are used. *Computer Aided Learning* (CAL) put emphasis on the learning process and thus is considered to be a cognitivistic view to the subject [76]. *Computer Assisted Learning* is considered to support traditional teaching methods while *Computer Based Learning* (CBL) is considered to replace a part of the curriculum [21].

Siviter and Brown proposes a definition for hypermedia based learning environment or *hypercourseware* to be a collection of topics that are a collection of educational activities [98]. *An educational activity* is a collection of activities such as reading a piece of text, looking at a picture, listening to a sound, looking at an animation, playing with a computer-based interactive device or following instructions to perform an assignment away from the computer [98]. The definition emphasizes the students active role on learning but lacks the fact that educational activities should be connected using hypertext links.

From the technical point of view the hypermedia based learning environment is a collection of studying material (textual, graphical) in the form of hypermedia, applications to be used in problem solving and cognitive tools. Jonassen defines *cognitive tools* to be tools that support, guide and broaden the learning process [59]. An example of a cognitive tool is a *concept map* that is a graph of the key concepts of the subject matter such that the nodes represent the concepts and edges between

the nodes represent relationships between the concepts [92].

Finally, *hypermedia based learning environment for mathematical sciences* consists of mathematical textbook and a mathematical dictionary as a hypermedia, mathematical tool programs and computer algebra systems and cognitive tools (for example concept maps) to support learning in mathematics.

What kind of theory is needed in order to support learning in hypermedia based learning environments? There are generally two different frameworks for the research of learning. According to behaviorism the students activity used to be the most important factor (stimulus - response). Knowledge is seen as a block which can be transferred from teacher or from the CAL package to the student. Students learn a lot of facts and they can apply their knowledge to similar problems as they have learned.

According to Jonassen, learning is thinking and thinking is activated by educational activities [59]. In *cognitive learning theory* learning is seen as a process of knowledge construction in which students develop a mental model or *orientation basis* of the learning subject [41]. As the learning process progresses students test the model and try to find its limitations. Information becomes knowledge when students append new information to their previous knowledge. This is why students can later apply their knowledge to a very wide range of problems. The cognitive learning theory seems to be the suitable framework to describe learning in a hypermedia based learning environments [60]. Hypermedia supports three general educational activities: information seeking, knowledge acquisition and problem solving [60], [93].

Hypermedia supports *information seeking* by providing a means to organize and present the same information in alternative ways. From the cognitive learning theory point of view, learning can be seen as the reorganization of knowledge structures or *schemas* [60]. The schema are connected by associations and are organized in the form of a network in the human mind [60]. The structure of the information in the hypermedia is nonlinear and is based upon associations. This is why hypermedia can be used to model the structure of information in the human mind and why hypermedia supports *knowledge acquisition* [60]. Hypermedia supports *problem solving* by providing tools, tutorials and analogies to solve the problems [60], [93].

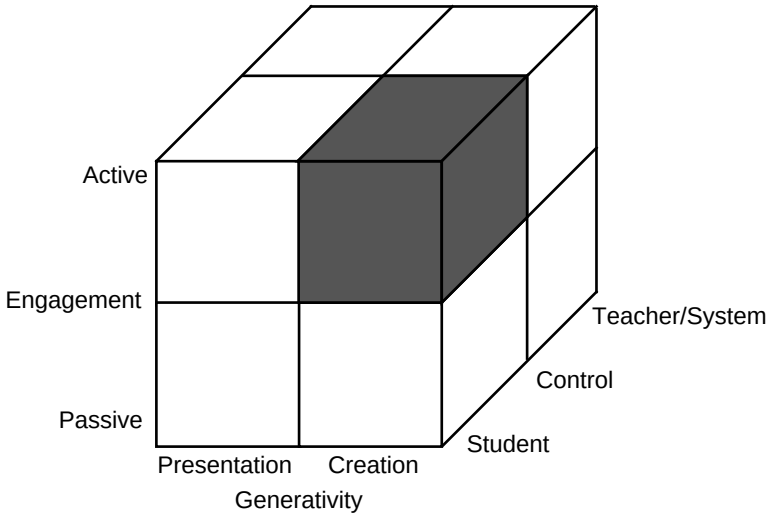

Figure 1.2: General framework for computer-based learning.

A well-known model for hypermedia that effectively supports learning is presented by Hammond [53] and Jonassen [59]. The model describes the general framework for computer-based learning in the form of a cube (figure 1.2). The three dimensions of the cube are engagement, control and generativity. Engagement represents the students role as an information processer to be active or passive. Control refers to the control of the learning process. It can be controlled by the student or the hypermedia system. Finally, generativity describes the students activity. The student can be an observer (presentation) or an author (creation). A hypermedia learning environment that effectively supports learning can be described as an environment where the students have the control and their role is active creator of the information i.e. the shaded octant in figure 1.2.

Siviter and Brown set up six requirements for hypercourseware: flexible learning, adaptability, modular structure, guidance, integration with other resources and exploiting hyper systems [98]. Flexible learning means that hypermedia based learning environments should be available in many learning situations such as in lectures or in self-study. The hypercourseware should also be easily adapted to the needs of different teachers and students. Modular structure of hypercourseware supports easy access to the learning material. Guidance supports the learning process by providing help in the usage of the software and by showing the students meaningful ways to

study the subject matter. Hypercourseware should also include references to other resources such as other CAL material and applications. By exploiting hypermedia systems it is possible to satisfy all other requirements [98]. These requirements are similar to requirements presented in this thesis for hypermedia based learning environments for mathematics.

Chapter 2

Related Work

In this chapter, the history of hypermedia, computer aided instruction (CAI) and computer aided learning (CAL) is reviewed. Early hypermedia and learning systems are discussed in 2.1. A few widely used hypermedia systems are discussed in 2.2. In 2.3 mathematical CAL software is discussed and evaluated. The purpose of this chapter is to indicate that there is no hypermedia based learning environment for mathematics available that would fullfill the requirements presented in previous chapter.

2.1 Early systems

Teaching and learning has been one of the first application areas for computers. Many educational systems have been designed and built. The motivation to use computers in education has been the believe that they can make learning more efficient and cheaper, at least, in the foreseeable future [85]. It has been seen that this is not true: the making of a lesson in CAI system has been significantly more expensive than conventional classroom teaching. Good surveys of early and current hypermedia systems are [13] and [29]. In Finnish, Suutarinen has made an extensive bibliographic survey of hypertext and hypermedia [102].

2.1.1 PLATO

A good survey of the history of CAI can be obtained from [85]. One of the first educational systems was PLATO (Programmed Learning and Teaching Operation) [88]. PLATO lessons were in general of drill and -practice, tutorial, inquiry, dialogue, simulation, computer games or problem solving. Special hardware, so called plasma display, was needed in order to view the courses. In addition to plain text, on plasma display it was possible to show raster graphics [88]. The PLATO terminal was also touch sensitive and included an audio device that could store up to 22 minutes of speech or music [88]. PLATO terminals were networked so that students were able to communicate with each other.

The PLATO has been used widely in universities and companies around the USA and Europe. In Finland, PLATO was used in the University of Tampere during 1979-1980 [74]. It was seen as an important innovation in the area of CAI and no negative facts were found during the testing. However, the effort required to produce material for PLATO was still quite high, approximately 15-200 hours per one hour lesson.

Apparently PLATO disappeared from the market when newer, innovative systems arrived. However, it has strongly influenced the development of other CAI systems such as COSTOC [101].

2.1.2 MEMEX

The first proposed hypermedia system was the MEMEX described by Vannever Bush [24]. He outlined MEMEX to be a desk with translucent screens, keyboard and storage for microfilms. MEMEX was able to associate documents to each other and users could easily add their marginal notes and comments to the documents. In principle, these are basic structures of hypertext links. MEMEX was never actually implemented but it has influenced to hypermedia systems implemented later.

2.1.3 Augment/NLS

One of the first implemented hypermedia systems was Augment/NLS developed by Douglas Engelbart. Augment was a project that developed tools for office automation and text processing [84]. As part of the project the NLS system included documentation produced by the Engelbart research group, specifications, plans, designs, programs, documentation, reports, memos, bibliography and reference notes [29]. Augment/NLS had hypertext features such as a database of nonlinear text and referential links.

2.1.4 Xanadu

The concept hypertext was invented by Ted Nelson [29]. He was developing a hypertext system Xanadu, that would collect all the worlds literary corpus online to form a *docuverse*. Any user could use existing material and pay royalities to the copyright holders of the original document. Once a document is saved to the docuverse only changes to the document can be saved. Original documents do not change and any version of the document can easily be recovered.

Xanadu has been developed since 1960 and an implementation called Xanadu Light exists today. It is further developed by the Australian company Serious Cybernetics under a name Xanadu Australia [87].

2.2 Hypermedia environments

In this section a few hypermedia environments are reviewed. Some of the systems presented here are succesfully used in education, so they could also be described as hypermedia based learning environments. However, in most cases the primary application area for these systems has not (yet) been education or computer aided learning.

2.2.1 Intermedia

A well known hypermedia system is Intermedia developed at Brown University's Institute for Research in Information and Scholarship (IRIS) between 1985 and 1990 [49]. Intermedia is a multiuser hypermedia framework where hypermedia functionality is handled at system level. Intermedia presents the user with a graphical file system browser and a set of applications that can handle text, graphics, timelines, animations and videodisc data. There is also a browser for link information, a set of linguistic tools and the ability to create and traverse links. Link information is isolated from the documents and are saved into a separate database. The start and end positions of the link are called anchors.

Intermedia supports learning by organizing the information in many ways. For example, timelines are a natural way to present historical events. Intermedia was implemented under A/UX, a Unix operating system for Macintosh computers, that has not been a very popular operating system for Macintoshes. It is not very likely that students can run Intermedia on their home computers because of the AU/X.

2.2.2 World Wide Web

World Wide Web (WWW) is a global hypermedia system on Internet [20]. It can be described as wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents [57]. It was originally developed in CERN for transforming research and ideas effectively throughout the organization [57]. Through WWW it is possible to deliver hypertext, graphics, animation and sound between different computer environments. To use WWW the user needs a browser, for example NCSA Mosaic[1] and a set of viewers, that are used to display complex graphics (JPEG), animation (MPEG) and sound (WAV). NCSA Mosaic is currently available on X-Windows, Windows and Macintosh.

In addition to NCSA Mosaic, there are several other WWW browsers available. Among them are Netscape Navigator[2] by Netscape Communications Corp., Arena[3]

[1]See http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html

[2]See http://home.netscape.com/

[3]See http://www.w3.org/pub/WWW/Arena/

developed in CERN, and HotJava[4] by SUN Microsystems Corp. Currently Arena and HotJava are available only for UNIX workstations. Netscape is available to Macintosh, Windows and Unix operating systems.

Authoring WWW documents

WWW browsers read and display hypertext documents that are marked with HyperText Markup Language (HTML, see Chapter 3). HTML documents can reside on different computers on Internet, and a document is referenced by URL (Universal Resource Locator). URL is of the form `http://computer.org.country/doc.html` where `computer.org.country` is the name of the computer and `doc.html` is the search path to the document.

The current standard of HTML is HTML 2.0 which includes support for forms. By using forms, the WWW client can send information to the WWW server. Forms can include text fields, buttons, pop-up menus and check-boxes. Under development is HTML 3.0 that includes support for tables and mathematical formulas (see page 50).

HTML documents can be written using any text processing application. However, there are specialized HTML editors, that include shortcuts to HTML tags and provide WYSIWYG editing. Such an application is HoTMetaL PRO[5], for example.

Documents can also be converted from other formats to HTML. For example, Rich Text Format (RTF) documents can be transferred to HTML by using a converter RTFtoHTML (available for at least Macintosh[6] and UNIX). It generates an HTML document from the original RTF document and a set of picture files if the RTF document contained pictures. In the HTML document links are created to the graphics files. The graphics can be viewed on most environments if pictures are of the GIF format. A similar converter exists for LaTeX and TeX documents (latex2html) in UNIX [38]. In both cases the raw text can easily be converted with converters and the quality of the converted text is quite good, but mathematical

[4]See http://java.sun.com/

[5]See http://www.sq.com/

[6]ftp://ftp.ncsa.uiuc.edu/Mosaic/Mac/Related/rtf-to-html-converter-275.hqx

notation, such as subscripts, superscripts and Greek alphabet are not supported by HTML 2.0. Formulas and other mathematical notation have to be included to the HTML file as GIF pictures. The quality of the final output depends on the WWW browser and the size of the selected font.

The links have to be written to each HTML file explicitly. This makes the authoring of large hypermedia material difficult. For example, a WWW course could contain several hundred HTML files and each file could contain approximately 10 links. Such a huge amount of links would be impossible to maintain by a human.

WWW browsers

Netscape 1.1 supports some HTML 3.0 features (tables) and has an interesting programming interface, that make it possible to develop external viewers that can communicate with Netscape. Moreover, Netscape has announced that it will incorporate Macromind Director Player Software to their browser. In the future, Netscape is also capable of displaying Adobe Acrobat documents (pdf documents) and applets written in Java language.

Arena is an experimental WWW browser developed in CERN. It supports HTML 3.0 and thus is able to display mathematical formulas and tables.

HotJava extends the WWW concept by making it possible to add links to small executable programs called applets. Applets are written in the Java language and are executed by the Java interpreter in the client computer i.e. in the users workstation. This makes it possible to make dynamical documents to the WWW. Applets can display animation, draggable 3D models or play sound. Java is an object-oriented programming language similar to C++.

WWW in education

There is quite a lot of educational material in WWW. Mainly these are educational resources such as programming code libraries or documents to be printed such as PostScript files. However, courses on different topics have also started to appear on WWW. The advantages of using WWW are that it is available globally and WWW documents can be read in all major computing environments. The disadvantages are

that often it is slow to open WWW documents because Internet has limited bandwidth - especially when using WWW at home with a modem. WWW documents are also difficult to author; there is a need for better authoring and conversion tools for WWW.

2.2.3 MathCAD and MathBrowser

Recently, the Mathsoft company has announced MathBrowser[7], a WWW browser that can display HTML and MathCAD documents. MathBrowser has a computational engine and interface similar to MathCAD, allowing the student to edit MathCAD documents through Internet.

The MathCAD runtime engine is used to distribute a collection of Schaum's outline series in electronic form [40]. They contain the same material as in the corresponding Schaum's outline series book. In addition to the text, it contains hypertext links from the table of contents to subchapters. There are links from subchapters to other subchapters or to examples but no links from concepts to their definitions. Interestingly, the student can change parameter values in example problems and see how the changes affect to the result. This is clearly a feature that supports learning and makes the mathematical text come alive.

2.2.4 HyperCard, Toolbook and MetaCard

HyperCard is a hypermedia authoring software for Macintosh computers [7], [10]. HyperCard consists a programming language (or a scripting language) HyperTalk and a set of painting tools for designing graphical user interfaces. It is based on a card-metafora. HyperCard application is called a stack or a collection of stacks. Each stack consists of cards and only one card is visible at a time in a stack. A card is displayed in a fixed size window. Hypertext links can be programmed by creating buttons and writing a HyperTalk script for the button. A script could execute an animation, present a dialog to the user or present a new card.

HyperCard was distributed with every new Macintosh and that made it rapidly

[7]See http://www.mathsoft.com/browser/index.html

a widely used prototyping environment for hypermedia applications. However, the disadvantages of HyperCard are limited search speed on large quantities of text, fixed card size in a stack, limited picture support and limited text field size [14].

In HyperCard editable text is displayed in a text field. The text field can be considered as a transparent layer in front of the card. However, HyperCard is unable to display mathematical formulas in a text field. That is why formulas must be included to cards as pictures. This makes the authoring of mathematical texts difficult, because the author is responsible for offseting the formulas in the card to correspond to the text in a text field. In addition, the contents of the card can not be scrolled while text fields are scrollable. The use of HyperCard in education is described in [4] and [62]. An example of HyperCard based learning material in mathematics is Mathematical MacTutor and Mathwise for Macintosh.

MetaCard is a similar application to HyperCard but it runs in Unix environments. MetaCard offers the ability to create and modify applications using interactive tools and a simple scripting language. Loimulahti has implemented an assessment software for first year university mathematics with MetaCard [78].

Intrestingly, HyperCard stacks can be imported to MetaCard. However there are incompatibilities with the HyperTalk and MetaTalk, therefore advanced stacks do not run without modifications.

Toolbook corresponds HyperCard in the Microsoft Windows environment. Toolbook has graphical tools for creating user interfaces, programming language OpenScript and an application programmers interface for the implementation of external code modules [12]. An example of ToolBook based learning material is Mathwise for Windows.

2.2.5 Hyper-G

Hyper-G is a multi-user, multi-protocol, structured hypermedia information system which runs as a client-server application on the Internet [5]. It was developed in Graz University of Technology, Austria. Hyper-G is intended to be a large-scale hypermedia system that could include tens of thousands of documents. In this aspect

it is similar to Xanadu or World Wide Web. The fundamental design principles of Hyper-G are collections, guided tours and searching [63]. Documents may be collected to collections, which may be a part of a collection itself. Collections form a directed acyclic graph in the document database [5]. Collections can be used to restrict navigation level, access rights or search scope. Collections in Hyper-G correspond to collections in the Dexter Model.

Guided tours are paths through the documents created by another user (for example topic specialist or teacher). Tours can be seen an important feature on a large-scale hypermedia system, especially on educational hypermedia systems [101].

Hyper-G documents may be searched by attribute (author, title, keywords) or by content. Documents are automatically indexed upon insertion into the database [5].

Hyper-G documents may be viewed with a special browser (Harmony for XWindows and Amadeus for MS-Windows) or with any WWW browser. If a WWW browser is used then Hyper-G documents are converted to HTML documents on the fly [5]. Obviously, the possibility to use WWW browsers extends the number of potential users considerably. Harmony uses external viewers to display text, images, MPEG films, audio, 3D scenes and PostScript. Any external viewer can be replaced with another external application capable of displaying such documents [5]. Harmony includes a graphical browser that shows collections in a tree structure. The Harmony Text Viewer uses SGML parser to display HTML documents and Hyper-Gs HTF (HyperText Format) documents.

An interesting fact in Harmony is its ability to adapt to the user language. Users can set up a list of preferred languages and in the case of multi-language documents, the document is displayed using the preferred language. In case of educational material, multi-language documents makes localisation of documents simple.

Mathematical formulas can be displayed in Harmony using PostScript or bitmap pictures in TextViewer. However, formulas may not be correctly aligned to the line. Future versions of Harmony will support HTML 3.0 which includes formula support.

Graphical browser for collections and the ability to draw a map of incoming and outgoing links of a node makes Hyper-G valuable for educational purposes. However,

there is not very much educational material available in Hyper-G servers. Currently there is only one Hyper-G server in Finland, in the University of Joensuu.

2.2.6 Adobe Acrobat

Adobe Acrobat is a software package that can be used to create portable document format (pdf) documents i.e. read-only documents that are readable in Unix, Macintosh and PC environments using Acrobat Reader[8] (that is public domain) [3]. Pictures, fonts and text charasteristics are preserved even if the original font is not present in viewer system. This is possible using Adobe Type Manager and special font templates.

Authoring pdf files is simple. A pdf-file is created using special printer driver that "prints" the document to a pdf-file. The pdf-file can include hypertext links to other pdf-files or to any other document. Links are created explicitly by drawing the link anchor to the source and selecting the destination. The destination can be a file or part of a file, for example a certain point in a drawing. Adobe Acrobat also includes indexing software that index a pfd-file automatically when the file is saved to a special directory or a folder.

Indexing makes Acrobat useful for publishing encyclopedias, reference manuals and similar material. Mathematical texts can easily be distributed in pdf form. There is a simple graphical browser that displays the table of contents in the form of fingernail pictures of each pdf page. However, for educational purposes a browser that would show the structure of links between nodes would be useful.

2.2.7 HyperTeX

A system called HyperTeX makes it possible to write HTML link definitions into TEX file by introducing a TEX macro `\href` [99]. TEX files are normally written with a text processing application and are then compiled to a device independent (dvi) format. Dvi files can then be viewed on the screen. For example,

[8] See http://www.adobe.com/Software/Acrobat/

```
Here is a link to \href{http://matwww.ee.tut.fi/}{hypermedia
laboratory}.
```

contains a link to a URL `http://matwww.ee.tut.fi/`. The link can be followed by using a special dvi viewer [99] that handles anchors as hot words. If an anchor is clicked then the dvi viewer calls WWW browser to open a specified URL. A WWW browser can be configured so that if the destination of a link is a dvi file then the dvi viewer is used to show the file.

HyperTeX system makes it easy to publish TEX documents on WWW. However, curretly HyperTeX capable dvi viewers exists only for Unix, Windows and Next computers. Even for these environments the viewers are on their early development stage [99].

2.3 Software for CAL

There exists a great amount of educational software for mathematics. For example, CTI center for mathematics and statistics in UK has published a catalog that contains over 850 titles [18]. Mannersalo has collated a catalog that contains over 240 software titles suitable for university level teaching of mathematics [79]. An extensive study of CAL in Finland is done by Lifländer [76].

Packages such as MathCAD, Matlab and Mathematica can be used in teaching as a computational tools. For example, Risku has used MathCAD in mathematics teaching and evaluated how students learn mathematics with computers compared to conventional teaching [95]. The results were promising. In almost every test the students gained better results that the teacher would have expected. Teachers summarized that subjects studied with computer and MathCAD were learned better than in the previous years with conventional teaching ([95], p. 98).

In addition, many books exists on educational use of Matlab and Mathematica [27], [43], [55], [81], [36], [89]. In this section we review only packages that have especially been made for computer aided learning and teaching.

2.3.1 COSTOC

COSTOC (COmputer Supported Teaching Of Computer science) has its roots in PLATO [101]. It has been developed at Graz University of Technology, Austria. The original idea was to convert a series of PLATO lessons to an Austrian videotext system. The main aim of COSTOC was to develop a large collection of CAI lessons on computer science. The implementation of authoring tools for COSTOC was not seen to be so important [101]. By the end of 1991, about 500 lessons were available in the COSTOC system.

In Finland, COSTOC lessons has been implemented by prof. Salomaa on topics such as "Computation and Automata" and "Cryptography and Data Security". Reported experiments of the use of COSTOC in Finland are for example from University of Kuopio, where COSTOC was used in the Introduction to Pascal course [69]. The course was evaluated by 20 students who considered it to be more lively than the book. The experimentation showed that the use of COSTOC had a positive effect on the results of the final test [69].

COSTOC was extended by developing a GLSS (General Lesson Specification System) that was used to integrate user-defined Pascal programs into CAI material [101]. COSTOC lessons can also be converted to GLSS and vice versa. Later, HyperCOSTOC was designed to allow non-linear browsing and tours. It was never actually implemented but the ideas have been incorporated to Hyper-G [101].

2.3.2 CTI and TLTP

CTI is described as The mission of the Computers in Teaching Initiative (CTI) and its aim is to maintain and enchance the quality of learning and increase the effectiveness of teaching through the application of appropriate learning technologies [28]. CTI was founded in 1989 in Great Britain and today there are twenty CTI Centres around Great Britain. Each Centre provides a support and information service for its given academic discipline. For example, Centre for Mathematics and Statistics (CTICMS) was set up to support and promote the use of computers in teaching mathematics and statics at degree-level throughout the UK. CTICMS

publishes quarterly Newsletter "Maths&Stats" where mathematical CAL software is reviewed. CTICMS also organizes workshops and conferences on CAL in mathematics and statistics.

Teaching and Learning Technology Programme (TLTP) was started in 1992 with the aim of making teaching and learning more productive and efficient by using modern technology. Currently there are 76 projects funded in the TLTP. Especially, there are 6 projects in the area of Mathematics and Statistics. The main idea of TLTP is to produce course material suited to degree-level courses that is to be distributed through CTI Centes.

Examples of TLTP projects are "A Campus Wide Structure for Multimedia Learning" at University of Southampton [104] and a Transitional Mathematics Project at Imperial College, London [64].

2.3.3 Mathwise

Mathwise is a learning environment for learning mathematics. It consists a set of files that are learning modules, reference leaflets or resources. In addition, the learning environment may include conventional hypermedia material, such as Toolbook books, word processor documents or computer programs. There is also a mechanism to index and search files that belong to the environment [16].

A learning module corresponds to 5 hours of conventional learning (i.e. lectures and exercises) [105]. The Courseware Design Guidelines suggest that learning modules are authored using HyperCard in Macintosh and Toolbook in PC [105]. This makes the authoring of Mathwise modules difficult (as was discussed in the previous section) and it is expected that it takes about six months to convert an electronic material of a module to the required screen format [105].

Reference leaflets correspond to drop-in tutorials and resources library books and programs. Currently Mathwise is available only in universities in UK.

2.3.4 MathAssessor

Diagnostic tests are used for example at the beginning of university studies or after a course in order to check the level of knowledge and start the preparation for the final exam. The MathAssessor was developed for these purposes [17]. There are problems in creating a mathematical assesment environment: mathematical questions are difficult to code-up in mathematical notation and diagrams and it is difficult to judge whether an answer is correct [17].

The MathAssessor program was implemented as dynamic link library (DLL) for Microsoft Windows. The questions are prepared on a word processor capable of saving in Rich Text Format (RTF). In order to mark up questions and correct answers, RTF is extended by adding several new commands, for example `\qn` for question, `\answer` for correct answer and `\numeric` to show that the answer is in numeric format. These commands have to be written implicitly to the text. The MathAssessor contains two parts: display of the questions and expression editor. First part is actually a special purpose RTF reader that is extended to parse the added commands properly. The second part provides an expression editor, which the student can use when inputting an answer.

Because RTF files are slow to open, the MathAssessor can create so called quick question files. These are quicker to open, but can not be opened by a word processor [17]. Currently, the MathAssessor supports numerical checking of answers by substituting a range of values to variables and comparing a student's answer to the correct answer. Under development is a syntactic checker that can be used instead of numerical check.

In addition to numerical answers, there can be multiple-choice questions and "point-and-click" diagrams. In a multiple-choice question the student has to select the right answer from the list provided by the MathAssessor. Diagrams may contains hidden hot areas and the student answers to the question by clicking the diagram. A click in the hot area is the right answer. In addition, the questions can include hints or help messages that originally are hidden, but can be opened by "stretching" the grey line in the question [17].

The MathAssessor seems to be a useful tool for creating diagnostic tests and self-assesment. As a part of the Mathwise learning environment the MathAssessor provides added value to Mathwise modules. However, it would be nice to be able to do hypertext links between the MathAssessor and Mathwise modules.

2.3.5 Mathematical MacTutor

Mathematical MacTutor is a set of HyperCard stacks containing demonstrations, animations, quizzies, puzzles and historical information from different areas of university mathematics [19]. Currently, there are over 150 learning modules on topics such as Permutations, Gaussian Elimination, Matrix Transforms, Vector Products and Planar Graphs. Mathematical MacTutor is intended to be a supplement to ordinary teaching material. It should not replace any of the conventional modes of teaching [19].

Mathematical MacTutor could be described as a mathematical experimenting tool. For example, in Matrix Transformations learning module the student can enter a 2×2 matrix A and then perform an operation AX, where matrix X contains the points of a geometric figure. The original and transformed figure is drawn to xy-coordinates. It is possible to select predefined matrices that perform reflection, dilation and rotation. In addition to the experimenting module, there is a short theory page of matrix transformations. The theory pages do not contain any hypertext links. Instead, every learning module contains a cross reference button which allows the user to move directly to other modules covering the same area of mathematics.

2.3.6 Calculus reform and interactive calculus courses

In the United States there has been a great effort to reform the teaching of calculus. The reform was motivated by the feeling that calculus courses had become a collection of techniques that the students had to remember. Calculus courses did not meet the needs for they were designed [37] i.e. to develop student's understanding of mathematical concepts, to expose students to a broad range of problems and approaches for solving them, to help students develop an appreciation of what math-

ematics is, to help students develop precision in both written and oral presentation, to help students develop their analytical skills, and to help students to understand other mathematical texts and materials [35]. These principles characterise not only calculus but all mathematics learning.

The reform called "Lean and Lively Calculus" emphasises narrower but conceptionally deeper syllabus for calculus courses [37]. It includes many improvements for the teaching of calculus, for example it suggests the use of technology, group work and formal oral presentations [35]. In addition, elementary theoretical problems and non-standard, context-free problems could be used to help students understand the calculus theory [35]. Calculus courses could also introduce multi-step problems which may take a few weeks to solve [35]. The suggestions are clearly based on cognitive learning theory.

It is important to realise that technology is not the only component of calculus reform. The main impact of using technology in calculus courses is that it is possible to use complex, real-world problems for introducing mathematical concepts without focusing on computational matters [37]. The availability of computer algebra systems or numerical software packages makes it possible for the students to experiment with their ideas [71]. Successful experiences from the new calculus courses are reported for example in the ICTCM conference proceedings [22].

In general, the affects of the reform of calculus to the syllabus can be considered positive. When, as proposed in [35], technology is used to support new teaching methods the calculus courses fullfil the principles and aims of learning mathematics.

There are whole calculus courses implemented in computer form. For example, a package called Calculus&Mathematica uses Mathematica notebooks to present theory, examples and exercises [33]. Students can fully access Mathematica by typing in commands in Mathematica language. Each notebook introduces the theory of the topic, which is followed by examples.

The Transitional Mathematics Project[9] (TMP) at Imperial College, London has produced similar Mathematica notebooks to be used in calculus courses [64]. At the end of each notebook there is a set of exercises to be solved with pen and paper and

[9]See http://othello.ma.ic.ac.uk

evaluated by the computer. Some feedback is provided for the student ("you got 2 out of 4 correct"). Mathematical formulas are created with a special formula editor and transferred to notebooks as pictures.

This kind of material certainly has its place in the classroom. It offers the capabilities of computer algebra systems to students right from the beginning of their mathematics studies. The disadvantage is that students must learn the material in sequental order; it is not possible to choose an individual order of progress. Apart from the possibility to open and close notebook cells, there are no hypertext features in Mathematica.

The presentation of mathematical formulas is a common problem for all educational software. For example, Mathematica 2.2 displays formulas using a monospaced font (for example Courier) so that Mathematica is able to print formulas on ASCII terminals as well as on notebooks. Calculus&Mathematica [33] contains a special font to improve the quality of formulas. In TMP [65] formulas are created with a formula editor and transferred to notebooks as pictures.

2.3.7 Mathematical simulation environments

A mathematical simulation environment is a software package that includes tools for modelling and simulating physical systems in a computer. Simulation environment is clearly a cognitive tool and this is why it should be a part of a mathematical learning environment. Cognitive tasks that simulation environments provide are reasoning about systems, predicting system behaviour and explaining outcomes [31].

Working Model [67] and Simulink [103] are examples of professional simulation environments, that can be used to simulate systems driven by physical laws.

In Working Model systems are designed by drawing the rigid bodies and constraints (e.g., motors, springs, and joints) with mouse and defining forces between the elements in the system. Meters and controllers can be defined in order to plot the data and adjust the parameters. Working Model does not show the mathematical equations of the system to the user. This is a feature that could be used in learning the mathematics behind the physical systems.

In Simulink the user has to define the model mathematically using block diagrams. A block is described for example using transfer functions or differential equations. A block can also be a measurement device that can be used to monitor the behaviour of the system.

The simulation environment alone may not be adequate as an efficient learning environment. Often a theory is needed to explain how to model a physical system and how to analyse the results of the simulation. Lieslehto [75] has designed and implemented a software package for multivariable controller design that includes a numerical engine (Matlab), an expert system and a hypertext interface (HyperCard). The expert system monitors the actions of the user and the results of Matlab functions. Based on this information the expert system analyses and interprets the numerical results and advises the user how to improve the design.

Simulating mathematical models and visualising the simulation data greatly affects to the learning process as was shown in our experiments with TDP simulator (see Paper 1).

2.4 Hypermedia based learning environments

In this section we review a few systems that correspond to the definition of hypermedia based learning environment. However, this review is not complete because many other systems could be described as hypermedia based learning environments. For example, a collection or collections of educational documents in Hyper-G can be considered as a hypermedia based learning environment.

2.4.1 Mathematical hypermedia on CD-ROM

Until recently only a few courses in mathematics have been implemented as hypertext. One example is "A Simple Introduction to Numerical Analysis" [54]. The material is published on a CD-ROM that contains the hypertext material, animations and graphical tools. Another interesting course is "Animated Algorithms: A Hypermedia Learning Environment for Introduction to Algorithms" [46], that is a hypertext version of "Introduction to Algorithms" [30]. This CD-ROM contains

hypertext, several animations and QuickTime movies. These high-quality texts and animations add a new dimension to learning mathematics. To see a sorting algorithm in action gives the user a mental reference or orientation basis with which to relate the theory and implementation of that algorithm.

However, there is a problem related to the way the material is put together. For the reader the material seems to be a collection of cards or stacks that do not form an entity. This problem is partly due to a limitation in HyperCard; HyperCard is not able to treat mathematical formulas as written text as described in page 29. That is why they must be transferred as pictures. HyperCard can present pictures and text only on cards, not on scrollable text fields. The card itself is not in a scrollable window and only one card can be visible at a time. As a result, when a new card is turned on, all the previous information disappears with the old card. This can be seen in [46].

2.4.2 Interactive book on lie algebra

Research Institute for the Applications of Computer Algebra (RIACA) was founded by The Foundation Computer Algebra Netherlands (CAN) in cooperation with the Stichting Matematisch Centrum (SMC) and the Kurt Gödel-School in Linz (Austria) in 1993. RIACA intends to play a pivotal role in the research and development of applications of computer algebra in mathematics, science and engineering. One of the main projects in RIACA has been the project "Human Interfaces: towards interactive books" coordinated by Norbert Kajler. It is also a part of the ACELA project at the Centre for Mathematics and Computer Science in Amsterdam (CWI) coordinated by Arjeh Cohen. The aim of the ACELA project is to design and implement an interactive book on mathematics, that would join mathematical hypertext and computer algebra systems. For example, the student could change values of parameters and immediately see the effect on the result. A prototype interactive book will be on Lie Algebras. ACELA is divided into five subprojects:

- The architecture and design of an environment and authoring system for interactive books on mathematics.

- The presentation of mathematics in the system.
- Automated deduction and theorem provers.
- The production of text in Lie Algebra.
- Design and implementation of new Lie Algebra algorithms.

Important issues in designing the system is that it should have ordinary hypertext facilities, the mathematical structure of a theory should be presented as a graph and it should be possible for the users to define their own notation used to present mathematical concepts. It should also adapt to the level of knowledge and interest of the student [2]. It should be possible to include annotations to the text and do computations with the underlying computer algebra system within the context of the book. The document formats used in ACELA are LaTeX for the texts and Maple, Mathematica or Axiom for computer algebra systems. The internal presentation of mathematics are handled using OpenMath (see page 52).

2.4.3 Microcosm

Microcosm is an open hypermedia system developed at the University of Southampton [52]. It is open in the sense that it makes it possible to define hypertext links between documents of different applications. Documents in Microcosm contain no mark-up. Instead, link information is held in link databases or *linkbases* [34]. Links can be specific links, local links, generic links or text retrieval links (dynamic links) [34]. Specific links can be followed after selecting the anchor at a specific location in current document. Local links can be followed after selecting the given anchor at any point in the current document. Generic links can be followed after selecting the given anchor at any point in any document. Text retrieval links are dynamically computed based on the selected text in the current document.

External applications called viewers are an integral part of Microcosm environment. In general, there are three types of viewers in Microcosm. Fully aware Microcosm viewers allow users to make selections from displayed data and then to request an action from a menu of possible actions such as follow link, make link or

compute dynamic link. Fully aware Microcosm viewers may also display active areas as buttons. Partially aware viewers are applications that are extended to include a Microcosm action menu by using the application's own programming language. Examples of such a viewers are Microsoft Word and Toolbook. Unaware viewers can not display a Microcosm action menu - they can only open requsted documents.

Currently Microcosm is fully implemented as stand-alone version in Microsoft Windows environment, but versions for Unix and Macintosh are under development. A network version of Microcosm is also under development [56].

2.4.4 Distributed information system for mathematics

In Germany, it has been proposed to set up a distributed mathematical information system (DISM) that could be used for mathematical university departments, research institutes and mathematical laboratories in industry [32]. The information system would include electronic access to full texts of research reports and educational texts, to mathematical software and data collections and they would be available on Internet. Other proposed partners include museums with scientific-technical departments and scientific publishing houses. For example, museums could provide scanned ancient mathematical documents, visualization of mathematical models, contemporary computer art and mathematical experiments in the form of living books [32].

The project is an example of *digital library*. A digital library can be described as a distributed information system, that holds interlinked textual and multimedia documents [44]. Digital libraries support learning by making diverse information resources available beyond the physical space shared by groups of students and by bringing together people with different learning missions [80]. For example, students could collaborate with professionals living all around the world and actually publish a (digital) report based on scientific data sets collected by researchers. Students and teachers can easily be publishers as well as readers in digital libraries [80].

For teaching purposes, digital library offers electronic lecture notes as a supplement and possibly an alternative to the usual lectures and printed lecture notes.

Students also have a wider choice of available texts, since the same courses are often taught at different universities. According to [32], this will affect to the quality of electronic lecture notes. When fully implemented, DISM could bring educational departments nearer to industries where mathematics are applied by providing contact information and job offerings.

Digital libraries have revealed many technical problems related to information retrieval support such as the display of complex graphics and scientific data containing mathematical formulas, distribution speed and reliability [42],[80]. For example, because many electronic journals are distributed in PostScript or LaTeX formats, users have to process (compile LaTeX documents and print PS documents etc.) the documents after loading them to the local workstation. Extra effort is needed in order to produce a hypertext version of a document published previously in paper format.

2.5 Summary

Computer-aided learning and CAL packages are still developing along with technologies such as hypermedia and fast, world-wide computer networks. The development of WWW, Hyper-G, digital libraries and Distributed Mathematical Information System may be the next generation learning environments that include enormous information resources. Currently the CTI Centres in UK has provided exellent CAL material and more importantly, trained students and university personnel to incorporate CAL in university courses.

The features of software packages presented in this chapter are summarized in table 2.1 and table 2.2. Each row describes a feature or features in a hypermedia or CAL software. The features are derived from the requirements that were set to the hypermedia based learning environment (Chapter 1). For each software package reviewed, there is a column in the table. Not every software package discussed in this chapter is included in the table. Mathematical simulation packages were not included because they can not be considered as learning environments as such. For some cases, there were no demos available and so the software package was not

included in the table.

In a conclusion, the hypermedia environments that were reviewed, do not satisfy all the requirements that were set for a hypermedia based learning environment. The CAL software that were reviewed are often designed for a special purpose, such as different modules in Mathematical MacTutor. This kind of software is useful in supporting normal teaching, such as lectures and home exercises. However they may not cover as much theory as is needed in a complete university course. In addition, CAL systems often do not support hypertext. In many cases, the authoring of a CAL software is difficult and time-consuming.

Educational material based on mathematical tool software such as Mathematica or mathematical simulation environments does not support hypertext and so they do not satisfy requirements for hypermedia based learning environments.

Many hypertext systems as such do not support learning i.e. they do not include the possibility to use cognitive tools as a part of a hypermedia environment. Microcosm is based on the use of external applications (it can not display any documents by itself) and it provides dynamic links. For displaying mathematical documents Microcosm uses Microsoft Word that should be available in the user's computer.

WWW seems to be an attractive system for hypermedia based learning environments. However, authoring of mathematical documents is difficult since the authoring language HTML does not (yet) support mathematical formulas. The same problem exists in HyperCard, ToolBook and MetaCard: their authoring tools do not support mathematical formulas.

Hypertext features or *hypertext functionality* starts to appear in many applications [86], for example MathBrowser from MathCAD is a good example of this development. However, it is difficult to convert existing mathematical documents for example to MathCAD format if the learning material is written in some other text processing system.

	Mathematical MacTutor	MathWise	The Assessor	Calculus & Mathematica	TMP	Microcosm	HMLE
Link types	Table of contents	Table of contents	No hypertext links.	Expanding or collapsing a cell	Expanding or collapsing a cell	Specific, local, generic and dynamic links	Explicit and implicit links, collapsing sections.
Educational activities	Each stack contains simulations, browsing	Reading, browsing	Hinted exercises, graphs with hot spots	Mathematica actions, entering and altering expressions	Mathematica actions, entering and altering expressions	Browsing, activities in Microcosm aware apps.	Browsing, drills, numerical & hinted exercises, movies
Network/Standalone	Standalone	Standalone	Standalone	Standalone	Standalone	Standalone (Windows)	Standalone
Environments	Mac	PC Windows, Macintosh	PC Windows	Mac, Unix, PC Windows	Mac, Unix, PC Windows	Windows, (Mac & Unix)	Mac
Authoring	HyperCard Stacks	HyperCard Stacks, Toolbook Books	RTF documents	Mathematica Notebooks	Mathematica Notebooks	Uses documents created with other applications	RTF documents and Mathematica notebooks
Graphical Browser	No	No	No	No	No	No	Concept Maps
Availability	Commercial	Commercial (not available outside UK)	Available from CTI-Centres in UK.	Commercial	Available from Internet	Commercial	Free
Special fonts	Supports Macintosh fonts	Windows fonts	Supports all Windows fonts.	Supports installed fonts	Supports installed fonts	Supports all installed fonts	Supports all fonts
Formulas	Formulas as pictures	Formulas as pictures	Microsoft Equation Editor	Uses special font to display formulas	Formulas as pictures	Formulas in Word documents	Imports Word Equation Editor formulas as pictures
Student activities	Browsing, learning-by-doing	Browsing	Numerical exercises, hinted exercises	Entering and altering Mathematica expressions	Entering and altering Mathematica expressions, drills	Activities on helper applications	Browsing, drills, numerical & hinted exercises, movies

Table 2.1. Summary of features of selected CAL packages

ISBN 951-722-561-X (printed version)
Reprinted from the original source to pdf 21.9.2004

	Netscape	Arena	MathBrowser	HyperCard	Hyper-G	Adobe Acrobat	HMLE
Link types	Links written in document	Links written in document	Links written in document	Explicit links.	Explicit and implicit links saved separately.	Explicit links	Explicit and implicit links, folding links. Link database.
Inter-application communication	External viewers	External viewers	External viewers	AppleEvents and HyperTalk	External viewers	AppleEvent links to external applications	AppleEvents, MathLink
Network/Standalone	Network & standalone	Network & standalone	Network & standalone	Standalone	Network	Standalone	Standalone
Environments	Mac, Windows and Unix	Unix	Windows	Mac	Unix, Windows	Mac, Windows, Unix	Mac
Authoring	HTML language Converters from RTF and LaTeX	HTML 3.0	MathCAD documents	Graphical tools	HTML, HTF and multi-language documents	Uses documents created with other applications	RTF documents and Mathematica notebooks
Graphical Browser	No	No	No	Fingernail pictures of visited cards	Document hierarchy, local link maps	Fingernail pictures of pages in table of contents	Concept maps
Availability	Free for academic users	Free	Free browser, commercial MathCAD is required for authoring	Commercial authoring software, free runtime version	Free for academic users	Commersial authoring software, free runtime version	Free
Special fonts	No special font support	Symbols	Supports MathCAD fonts	Supports all fonts	PostScript support	Supports all fonts - also fonts not installed in viewer	Supports all fonts
Formulas	GIF pictures	HTML 3.0 tags	MathCAD formulas	No support in text fields	PostScript, pictures	Supports formulas	Imports Word Equation Editor formulas as pictures
Student activities	Browsing, filling forms	Browsing	Browsing, altering MathCAD formulas	Browsing, graphical controls, etc.	Browsing	Browsing	Browsing, drills, numerical & hinted exercises, movies

Table 2.2. Summary of features of selected hypermedia packages

ISBN 951-722-561-X (printed version)
Reprinted from the original source to pdf 21.9.2004

Chapter 3

Document Markup Languages

In this chapter two topics are discussed: document markup languages and communication protocols between mathematical applications. The chapter provides background information to the conversion problem: how to convert ordinary mathematical text to hypertext format (RP 3). The solution to the problem is given in the next chapter. The discussion about communication protocols is an introduction to the problem how mathematical applications can be included into the hypermedia based learning environment (RP 6). The role of Mathematica and MathLink communication protocol in HMLE is discussed in Paper 3.

Bryan ([23], p.5) defines markup as: "Markup is the term used to describe codes added to electronically prepared text to define the structure of the text or the format in which it is to appear." There can be two types of markups: specific markup and generalized markup. Specific markup describes the format of the document (fonts, text styles etc.) whereas generalized markup describes the structure of the document (headings, citations etc.). For example, Rich Text Format (RTF) is a specific markup language and TEX, LATEX, SGML and HTML are general markup languages.

3.1 Standard generalized markup language

SGML is an international stardard (ISO 8879) for document markup. An SGML document contains a document type definition (DTD) and a set of elements that are defined in DTD ([23], p.20). Each element has a name and it can be used as a *tag*

in SGML document. For example, in figure 3.1 there is a simple SGML document, where the first line expresses the DTD, and <memo>, <from>, <para> and <to> are tags. The actual DTD is often in a separate file.

```
<!DOCTYPE letter PUBLIC "-//DTD Letter">
<letter>
<from> Jari Multisilta
<to> N.N
<para> See you tomorrow.
</letter>
```

Figure 3.1: Simple SGML document.

In general, DTD defines the names of elements that can be used in a document, the order of the elements, how often elements can appear in a document, which elements can be omitted and the contents of the elements (attributes and elements that can appear inside an element, data) [106].

Mathematical texts often have a clear structure. They consist of axioms, definitions, theorems, proofs and examples. These could be used as elements in mathematical text (figure 3.2).

```
<DEF name=Matrix>A matrix is a table of...</DEF>
<THRM name=Cauchy-Schwarz>Let...</THRM>
<PROOF name=Cauchy-Schwarz>...</PROOF>
<EXMPL name=Vector norm>...</EXMPL>
```

Figure 3.2: SGML tags proposed for mathematical texts.

<DEF> corresponds to the definition, <THRM> corresponds to the theorem, <PROOF> corresponds to the proof of the theorem and <EXMPL> corresponds to the example. The tags presented in figure 3.2 could be used to implement implicit links and concept trees by analysing <DEF> tags and their contents. The definition of a root concept is analysed and mathematical concepts are searched from the text

of the root. Found concepts could be inserted into the concept tree or links to their definitions could be made.

Mathematical formulas introduce problems in SGML. Formulas can be included as TEX in SGML. There have been efforts to produce a DTD for mathematical formulas (see for example [106], pp. 207-216). However, van Herwijnen argues that a design of a logical DTD that describes the structure of mathematical meaning is as impossible as it is for natural language ([106], p. 213). This is why the current harmonized mathematics DTD describes only the presentation or visual structure of formulas [106].

3.2 Hypertext markup language

Hypertext Markup Language (HTML) is an SGML based markup language for WWW documents. HTML DTD[1] was originally defined in CERN where WWW was developed. HTML DTD is implemented in WWW browsers so that users can change text font and size of certain DTD elements.

The current version of HTML language is 2.0. It includes basic document formatting, linking, anchoring mechanisms and forms. Forms are interactive WWW pages that can consist of buttons, pop-up menus and editable text items. The user can submit the form back to the server that processes the contents of the form. For example, users could order a book by entering their contact information and credit card number to a form. Under development is the HTML version 3.0 which supports mathematical formulas, floating figures and tables [20]. Mathematical elements in HTML 3.0 are "inspired by TEX", for example $y = ax^k + bsinx$ is expressed in HTML 3.0 as

```
<math>y = a x^k^ + b sin x</math>
```

and in TEX as

```
$y=ax^{k}+b sin x$.
```

[1]See http://www.w3.org/pub/WWW/MarkUp/MarkUp.html

In HTML 2.0 mathematical formulas have to be included as pictures within the text.

3.3 TEX and LATEX

TEX is a document preparation system designed for high-quality typesetting, especially for mathematical text [68]. TEX language is a complex language containing hundreds of commands. For example, {\bf some text} is a TEX control sequence expressing that "some text" is to be printed in boldface. TEX commands can be grouped to macros that perform complex tasks. LATEX is a set of TEX macros, that simplify typesetting by introducing commands such as \chapter, \section or \subsection [72]. With LATEX it is possible to write fully structured documents. In order to view a TEX or LATEX document, it has to be compiled to dvi (device independent) format. Dvi documents can then be printed or viewed in many computing environments using a dvi viewer application.

It is staightforward to define a markup in LATEX that supports hypermedia. For example, Kivelä has suggested a stucture to be used to express a link in LATEX, such that an outgoing link is expressed as \outlink{source}{destination}, where source and destination are link anchors in the source and destination nodes [66]. It is then the matter of the LATEX viewer (dvi viewer) to interpret these link definitions (see HyperTeX discussed on page 32).

The structure of mathematical texts can be expressed in TEX and in LATEX. It is possible to write macros for definitions, theorems etc., so that TEX or LATEX automatically numbers each definition and theorem correctly.

3.4 Rich text format

Many word processors can open and save documents in RTF (Rich Text Format). RTF is a document markup language for text formatting, pictures and formulas. RTF files are plain text (ascii) files (figure 3.3). RTF is defined by Microsoft and is described in [83].

The difference between SGML and RTF is that SGML describes the structure of a document, whereas RTF mainly describes the physical characteristics of the text (text face, size, etc). However, RTF also includes certain tags that describe document stucture. The author can define a set of styles for the document (heading 1, heading 2, abstract, etc) that are written into the beginning of the RTF file and have a special tag in the RTF markup.

```
{\pard\plain \f15007 This is simple RTF document with subscript
x{\dn4 i} and {\i italics}.\par}
```

Figure 3.3: Example of RTF markup.

In RTF it is not easy to express the structure of mathematical texts. The author could create styles for definitions, theorems etc. and these styles could then be used to create structured hypertext documents.

In RTF files produced by Microsoft Word 5.1 there can be so called objects - links for example to Exel data or to objects created by Equation Editor. The RTF file presents a mathematical formula in two different ways: as an OLE (Object Linking and Embedding) object and as a hexadecimal coded picture. The picture information is easy to translate to a visible format. The drawback is that the logical structure of the formula is not preserved.

3.5 Communication protocols

In addition to the display of mathematical formulas in hypermedia systems, formulas could be used as live elements of a document. For example, a user could change the value of a parameter in a formula and compute it again using a computer algebra system. For this purpose, a communications protocol between hypermedia system and computer algebra system is needed. There are a few such protocols, for example Multi Protocol (MP), OpenMath and MathLink. General communication protocols are also researched in the Interspace project at the University of Illinois, where a large-scale digital library is to be built [97].

The MP is designed for integrating symbolic, numeric, graphics, document processing and other tool applications into a single distributed problem-solving environment [48].

The OpenMath consortium is an international group of researchers designing a protocol for exchanging mathematical information between applications [1]. OpenMath is a successor to MP since the authors of MP are contributing to the OpenMath consortium [48]. Waterloo Maple Software and REDUCE have been used to demonstrate the OpenMath proposal [100]. For example, using OpenMath a general purpose computer algebra system could call a specific purpose application to execute an algorithm implemented only in this application.

OpenMath tries to preserve semantic information in addition to the structural information of the formula. For example, TEX describes only the visual appearance of a formula, not the semantic structure of the formula. Similar visual representations of mathematical formulas have been planned for SGML. OpenMath will include SGML compatibility [1], so that OpenMath objects can also be included in SGML documents.

MathLink is a communications protocol for exchanging Mathematica expressions and data between Mathematica and external applications [109]. The difference between MathLink and OpenMath is that MathLink does not define the semantical structure of a formula. Instead, MathLink communicates by sending and recieving Mathematica expressions as strings [109]. The advantage of MathLink compared to OpenMath is that it is already working in the current versions of Mathematica and it can be used in Macintosh, Windows and Unix environments. Paper 3 concentrates on descibing the use of MathLink in HMLE.

3.6 Summary

In this chapter, document markup languages and selected mathematical communication protocols were explored. SGML and HTML are general markup languages that describe the logical structure of the document. They can also be used to describe the structure of mathematical documents. At present, however, it is not easy

to describe mathematical formulas using SGML or HTML.

TEX and LATEX are ideal markup languages for mathematical texts. However, current hypertext systems do not provide support to TEX and LATEX documents. LATEX documents can be converted to HTML as discussed in Chapter 2. Another possibility is to use special dvi viewers for displaying TEX and LATEX documents (see HyperTeX in Chapter 2). Such viewers are still in the early development stages.

The Department of Mathematics (TUT) material had no markup that could have been used in the conversion process. RTF was selected because material (i.e. original lecture notes) had been written with Microsoft Word (Paper 2) that can save documents using RTF. Manual conversion of the existing lecture notes to another format, like TEX or LATEX would have contradicted the requirement of easy authoring of hypermedia based learning environments (Chapter 1).

Chapter 4

Textbook Conversion to Hypertext

In many cases the development of a hypermedia course is based on existing textual material (text books or lecture notes). This is why there should be easy-to-use tools for converting existing material to hypertext format. In this chapter the document conversion problem is discussed (RP 3). Earlier experiences in converting documents to hypertext format are presented. RTF Reader is outlined as a solution to convert mathematical documents to hypertext. The implementation of RTF Reader is discussed in the following chapter.

Traditional documents may be transformed into hypertext by creating nodes and links based on the logical markup of the document (chapters, sections, footnotes etc.) and then adding links not evident in the document markup [91]. First-order hypertext includes links that are explicitly marked in the original text. These include links from table of contents to section headings, links to bibliographic references and to footnotes. Second-order hypertext includes links that are not explicitly written in the original text. These types of links can be computed based on word patterns in the text [91].

4.1 Previous experiences

Rada reports on the conversion process of a textbook called "Hypertext: from Text to Expertext" to the hypertext form. The book was available with standard Unix document markup, troff. The printed version of the book has about 250 pages and includes about 150 figures. The book was converted to four hypertext environments, namely Emacs-Info, HyperTies, Guide and SuperBook. The text was divided into text blocks that corresponded to the subsections in the book and the logical markup was replaced with the markup used in the hypertext system. For the second-order hypertext, word patterns were used to create links between text blocks ([91], p. 305). In addition to this, alternate outlines were added manually.

Frisse describes the conversion of a part of a manuscript for medical textbook into an experimental hypertext handbook using Xerox NoteCard [45]. NoteCard is a similar hypertext environment to HyperCard since it uses card metafora to display data on a fixed size card. The medical handbook included hierarchically organised text. A text-parsing program was used to convert flat text files into a hypermedia document by exploiting the document identifiers to be parsed. Each card was indexed by creating inverted indexes by eliminating stop words, removing suffixes and retaining word roots as indexes into the card ([45], p. 250). In addition to indexing, for each card a card weight was computed so that the user could identify the best set of cards for browsing a requested topic.

Inside Macintosh, the Macintosh programmers reference was converted to hypermedia by Bechtel [14]. It was implemented as a HyperCard stack and distributed on a CD-ROM. The most important design principle was to maintain the logical structure of the printed material. A fast boolean search software was added to HyperCard in order to provide full text search capabilities. Because HyperCard text fields may contain 32,000 characters at most, the chapters were divided into several cards. For browsing, a table of contents, index, subject list and chapter list were provided. No explicit links were created to the main texts. This was the main critisism from the users of the CD-ROM version of Inside Macintosh [14].

Chignell and co-workers have designed the HEFTI model for converting textual

documents to hypertext. In HEFTI, the conversion process is divided into six subtasks: text preparation, node preparation, indexing, link creation, organization and link refinement [26]. In text preparation, the original text is converted into electronic form and marked up so that each chapter, subchapter, section etc is numbered and tagged. In the node preparation phase the material is divided into nodes. The next step consists of indexing each node. The link creation involves the creation of links based on indexed nodes. Links are created using document similarity [26]. The organization step involves the organization of documents using hierarchical model or using important nodes as landmarks. The organization step is performed by generating more links between nodes. The final step, link refinement checks that there are enough links coming in to and out from every node. The output of the process is a text file constructed in a specified hypertext markup format. The HEFTI model has been used to convert a 150 page book into a hypertext [26]. The authors succeeded in completing the conversion within one long working day. The results from the usability analyses suggested that hypertext produced by HEFTI is comparable to many manually authored hypertexts [26].

4.2 Experiences at TUT

In the department of Mathematics at Tampere University of Technology (TUT) part of the lecture notes have been written with Microsoft Word. For example, lecture notes for the Matrix Algebra course consist 146 printed pages. In addition there are over 60 pages of exercises and several Matlab scripts used in laboratory demonstrations. It was decided to convert this material to hypertext.

HMLE was designed so that it can import RTF files (Paper 2). Because RTF files are slow to open, a new binary file format (CNVR files) was designed to support faster opening of nodes. RTF Converter was implemented in order to convert RTF files to CNVR files.

Converting a file or a directory is a sequence of opening of the specified RTF file in a new window, saving a CNVR file and closing of the window. Converting a directory is applied to all RTF files in the directory (nested directories are not

traced). In general, RTF Converter implements RTF Translator algorithm presented in page 59.

The authoring of the HMLE courses are discussed in Paper 4 and an overview of the authoring procedure is presented in figure 4.1. In general, it corresponds to the HEFTI model.

1. Create directory structure in HMLE server and divide the original textual material to nodes corresponding to the directory structure.
2. Convert RTF files to CNVR files
3. Import CNVR files to HMLE server and create keywords for each node
4. Create explicit links

Figure 4.1: Conversion of RTF documents to hypertext.

4.2.1 Dividing the text into nodes

Since RTF only marks the structure of the text if the author has used specified styles, the division of the text to nodes have to be done manually.

The author has to design a directory structure of the hypertext material and divide the text into nodes. For mathematical courses nodes could be for example subchapters, definitions, examples, exercises and theorems. The structure of the node database is explained more precisely in Paper 2.

The division of the text to nodes corresponds to the text and node preparation steps in the HEFTI model. The nodes are then converted from RTF to CNVR format and CNVR files are imported to HMLE. This step corresponds to the indexing step in HEFTI.

4.2.2 Generating hypertext links

In order to produce hypertext, considerable number of links should be made. In the case of the Matrix Algebra course, there are about 70 subchapters, 80 definitions

and 70 exercises, each containing an average of approximately ten hotwords. It would be quite difficult to do all the links manually. This is why implicit linking is an important feature in HMLE. Implicit links are discussed on page 73.

In addition to implicit links the author can make explicit links using Link Tool described on page 72. The making of explicit links corresponds to the organization step in the HEFTI model.

4.2.3 Evaluation of the conversion process

The disadvantage of the converson process described above is that the division of the text to nodes has to be done manually. It is difficult to overcome using RTF. However, the conversion of the text to a structured markup would have required even more manual work. This was seen for example when the Matrix Algebra course was converted to HTML (Paper 5). In our experience the production of the WWW version involved much more work that of the production of the HMLE version.

4.3 Algorithm for RTF Translator

RTF file consists of unformatted text, control words, control symbols and groups [83]. A control word is of the form

```
\LetterSequense[NumericParameter] <Delimiter>
```

For example `\up6` is a control word for superscript of six pixels. Control words affect groups which are enclosed in brackets {}. Groups can consist of control words, groups and ordinary text. For example, $(x + y)$\{\up6 2\} would print as $(x + y)^2$.

RTF file has to be processed by an RTF Translator. An algorithm for simple RTF Translator is presented in figure 4.2 (c.f. [83]).

The first step reads and processes the definition tables, such as font, style and color tables. After the tables the translator starts to process the main text by reading one character at a time. If the character is the opening of the group, the current syles should be saved to the style table and an empty (default) style should

be initialized. If the character is the closing of the group, the translator restores the style information from the previous group. If the character is a backslash the translator reads the RTF control word and parameters, and changes the current style to correspond to the RTF control word.

```
Step 0. Process the definitions tables
Step 1. Read next character
Step 2. Case character
        {: save the current state of the style in the style table.
        }:  restore the previous state from the stack.
        /:  proceed an RTF command.

        Else insert ordinary text into te handle

Step 3. Go to step 1
```

Figure 4.2: Algorithm for RTF Translator.

4.4 Summary

In this chapter, the solution to the conversion of mathematical documents to hypertext was presented. The conversion is semi-automatic: the author has to divide the text to nodes manually. It would be difficult to create the nodes automatically based on the RTF markup. However, hypertext conversion procedures often include manual work as was seen from the examples presented earlier in this chapter. The advantage of the conversion process presented here is that it creates the structures for implicit linking.

Chapter 5

Implementation of HMLE

In this chapter, the implementation of HMLE is discussed. Especially, RTF Reader and Exercise Maker provide solutions to the presentation problem of mathematical texts.The RTF Converter is a solution to the conversion of mathematical texts to hypertext. In order to understand the discussion of the implementation of HMLE the basic concepts of Macintosh programming are presented in the following section. Authoring tools and learning tools for HMLE are also presented.

5.1 Programming with Macintosh and HyperCard

Macintosh programming is based on event-loop applications that handle events generated by user actions. Menu selections are typical events. A part of the Macintosh system software is a toolbox that is a collection of managers, for example menu manager, window manager and dialog manager. Each manager defines a set of routines that can be called from applications in order to implement standard Macintosh applications. These routines are documented in the Inside Macintosh series published by Addison-Wesley [8] and in Macintosh Technical Notes published periodically by Apple Computer. Inside Macintosh and Macintosh Technical Notes are available in electronic form for example in Essential Tools and Objects (ETO) CD-ROM [11].

Applications in Macintosh send messages to each other by using Apple Events. Macintosh toolbox provides an interface for sending and receiving events. In general, applications that share events must agree on what kind of services they can ask for

each other and what are the actions of the events. For example, a text processing application could ask for a converter application to convert a file from one format to another by sending an Apple Event.

5.1.1 Programming the toolbox

The TextEdit package in Macintosh toolbox defines the standard way to handle text documents in Macintosh applications. TextEdit provides a basic data structures (`TEHandle`, `TERec` and `TEStyleRec`) and routines for editing styled text. For example, there are routines to insert and delete text to and from a `TEHandle`, set insertion point and set the text charasteristics such as font, font size and text style. Text in `TEHandle` can be measured (i.e. an exact pixel position of a character can be computed in window coordinates) and drawn using QuickDraw, a package for drawing in Macintosh. However, TextEdit lacks super- and subscripts and pictures.

Almost all data structures in Macintosh toolbox are referenced by *handles*. A handle is a pointer to the pointer - i.e. a double reference to an allocated memory block (figure 5.1). Memory space for data structures is reserved dynamically from the application heap. In order to allocate memory dynamically, Macintosh Operating System (OS) may move the location of reserved blocks in memory. This is why memory blocks are often referenced by handles. When Macintosh OS moves a block in the memory it updates the block's master pointer. The master pointer can never move in memory.

5.1.2 Programming external commands for HyperCard

HyperTalk is the programming language for HyperCard. It is an object oriented language in the sense that stacks, cards, backgrounds, fields and buttons are objects. Objects communicate by sending and handling messages - a HyperTalk program is actually a collection of message handlers. For example, a button sends a `mouseDown` message when it is pressed and the message is handled by the handler `on mouseDown`. The message is first passed to the button itself. If the button does not have a handler that can handle the message it is then passed to the card where the button belongs

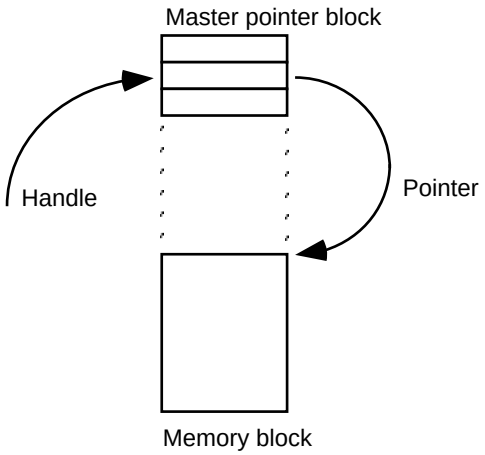

Figure 5.1: Referencing with handles.

to. If the card does not handle the message it goes to the background, stack, home stack and finally to the HyperCard application. This is called a hierarchy of HyperCard's objects [47].

HyperTalk can be extended by programming external commands, XCMDs, for example in C or Pascal. HyperCard includes a programming interface for XCMD's [10]. When calling an external function HyperCard passes parameters to the XCMD and it can return a value back to HyperCard using a special data structure `XCmdBlock`.

There is also a set of utility routines for handling HyperCard objects and strings such as `PasToZero` which converts Pascal strings to HyperTalk strings. XCMD interface has routines to set up external windows so that HyperCard knows which windows were created by an XCMD and can send messages to these windows. Messages that are sent to an XCMD window are for example `open`, `close`, and `update`.

5.2 HMLE architecture

In the design of HMLE it was defined that it should display mathematical documents saved in RTF (c.f. requirements for hypermedia based learning environment, Chapter 1). It should also provide hypertext link services, navigational aids and learning tools to support learning (Paper 2).

At the early stage of HMLE development, HMLE was built around a HyperCard

stack called HMLE Stack. Documents were displayed by RTF Reader in HMLE Stack. It was also the authoring environment for HMLE courses. HMLE Stack could open other applications in order to display nodes created with Mathematica, Maple or Matlab. Exercise Maker was used only for presenting interactive exercises. In the early version users interacted with HMLE Stack and Exercise Maker both of which had their own interface and navigational aids (i.e link palette in HMLE Stack and link buttons in Exercise Maker). The design and the architechture of the early version of HMLE is explained in more detail in Paper 2.

In order to make the user interface more consistent the HMLE was divided into a server application (HMLE Server) and a client (Exercise Maker) (figure 5.2). They communicate with each other using Apple Events. Students interact only with Exercise Maker which displays all CNVR documents (table of contents, main texts, definitions and exercises) and provides navigational aids for the students.

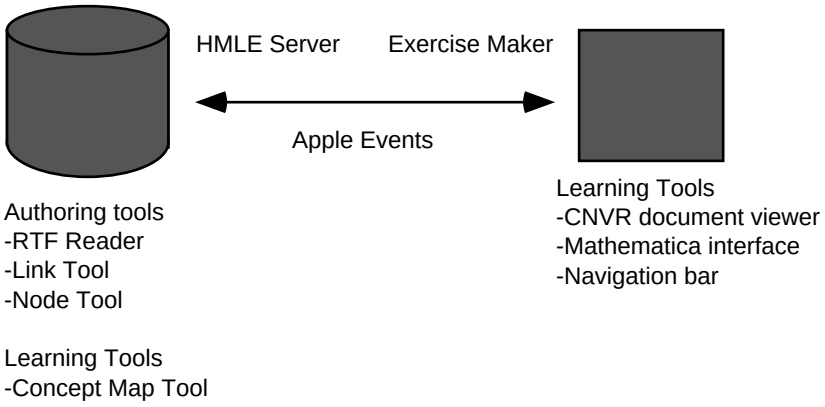

Figure 5.2: Client-server architechture of HMLE.

HMLE Server provides hypertext link services to Exercise Maker and it is the authoring environment of HMLE courses (Paper 4). HMLE server consists of RTF Reader, Node Tool, Link Tool, Concept Map Tool and node and link databases. Each node in the node database is referenced by a unique node ID assigned by the HMLE server.

HMLE server includes all the functionality of HMLE Stack. The difference is that students do not interact with HMLE server. Exercise Maker, RTF Reader and RTF Converter are all based on IDXTextEdit, an extended TextEdit package

developed for HMLE.

HMLE server clearly belongs to the storage layer in the Dexter Model. However, since HMLE server has information about the contents of the nodes (typed nodes, Paper 2) it includes features from the within-component layer also. Exercise Maker belongs to the run-time layer in Dexter Model because it is responsible for presenting the hypertext to the users.

Important elements in the Dexter Model are interfaces between run-time and storage layer (presentation specifications) and between storage and within-component layer (anchoring). The presentation specifications for a node is described as the node type in HMLE. Each node type can be viewed with a specific application. For example, QuickTime movies can be viewed by Movie Player, a standard application for QuickTime movies.

HMLE server implements resolver and accessor functions which belong to the storage layer in Dexter Model. For example, the resolver function is needed to create an implicit link from a mathematical concept to its definition. The actual definition node is accessed by the accessor function that maps node IDs to the file names.

5.3 RTF Reader

In HyperCard it is somewhat difficult to present large chunks of text with super- and subscripts and other mathematical notations (Chapter 2.2.4). RTF Reader XCMD was implemented to read RTF and CNVR files and to display them in a HyperCard external window (called the RTF window). The XCMD understands the subset of RTF that deals with fonts and styles, super- and subscripts (.\dn,.\up), pictures and overstrikes (.\ |o). It will ignore some formatting commands, for example page headers and footers, colors and footnotes. The XCMD runs under HyperCard 2.0 or later.

RTF Reader has hypermedia features for creating explicit links. Using Link Tool the author can create string-to-lexia links by activating a string in an RTF window and selecting the destination node from a dialog.

Data structure `WRefRec` describes the RTF window (figure 5.3). In the `WRefRec` data structure the `docTE` field contains a handle to the text and to its style information, `docVScroll` and `docHScroll` are control handles for vertical and horizontal scroll bars and `docClick` is special click loop routine field (Chapter 5.4.2).

```
WRefHndle = ^WRefPtr;    { Handle to my X-window data }
WRefPtr = ^WRefRec;      { Pointer to my X-window data }

WRefRec = record         { RTF window data. }
    docTE: TEHandle;     { Text-edit fields. }
    docVScroll: ControlHandle;
    docHScroll: ControlHandle;
    docClik: ProcPtr;

{ Properties }

    fileName: str255;    { Path to the file where the data is. }
    nodeID: integer;     { ID of the node. }
    kwStyle: style;      { The style for keywords.}
end;
```

Figure 5.3: `WRefRec` describes the state of RTF window.

RTF windows have special properties that can be set from HyperTalk. For these properties there are `fileName`, `nodeID` and `kwStyle` fields. The field `fileName` contains a search path to the actual node data. The field `nodeID` is the unique ID number of the node assigned by the HMLE Server. The field `kwStyle` includes an information of the style that is used for emphasizing certain words in the node text. For example, in RTF Reader it is possible to retrieve all words in italics by setting `kwStyle` to `italics`.

5.4 IDXTextEdit

In order to be able to display super- and subscripts, pictures and mathematical formulas in a window, the TextEdit in Macintosh toolbox had to be extended. Fortunately, the TextEdit package has been designed so that it can be extended easily by implementing a new data structure and providing routines to handle the new data structure. The extension to TextEdit manager developed in this work is called IDXTextEdit. RTF Reader, RTF Converter and Exercise Maker use IDXTextEdit routines for displaying mathematical documents in HMLE.

5.4.1 IDXTextEdit data structures

`TEHandle` is a handle for the TextEdit data structures (`TERec` and `TEStyleRec`) provided by the Macintosh toolbox ([9], pp.15-38). `TEStyleRec` has a field `teRefCon` that was used to save a handle to the `IDXTextEditRec` data structure (figure 5.4).

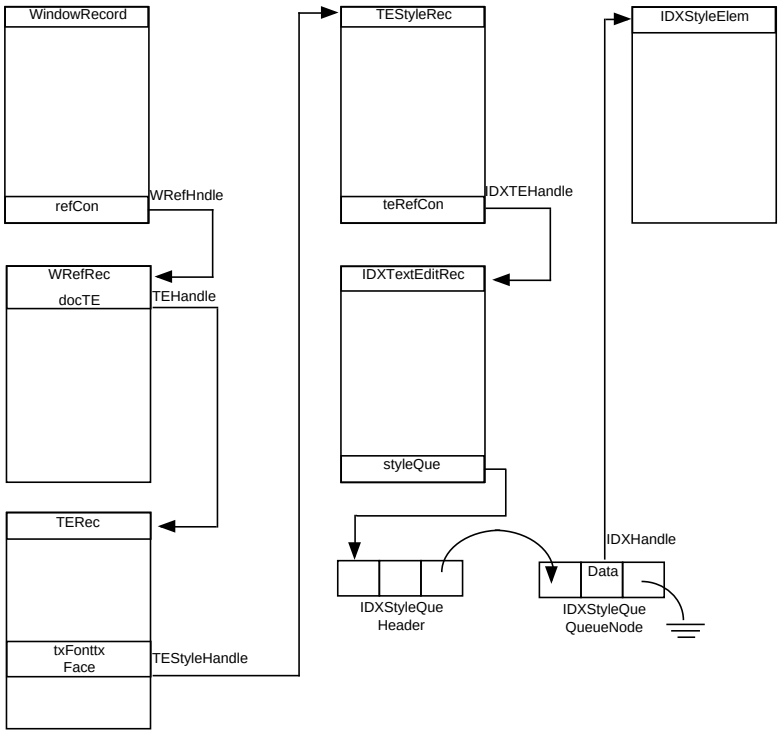

Figure 5.4: Summary of the data structures in the IDXTextEdit.

The main idea is to insert all (normal) text into `TEHandle` and keep information about which characters should be drawn to the super- or subscript positions and

```
IDXStyleElem = record
    startChar:integer; { Start character for our style run. }
    endChar: integer;  { End character for our style run. }
    indexPos: integer; { Super- or subscript position. }
    pict: picHandle;   { Handle to picture. }
    overLine: str255;  { Overline characters. }
end;

IDXPtr = ^IDXStyleElem; { Pointer to IDXStyleElem. }
IDXHandle = ^IDXPtr;    { Handle to IDXStyleElem. }
IDXStyleQue = queHndle; { Handle to the index style queue. }
```

Figure 5.5: Data structure for `IDXStyleQue`.

which characters represent pictures in `IDXTextEditRec`. McKenzie has presented a similar implementation for super- and subscripts [82]. However, his method has some drawbacks - it does not include pictures, and super- and subscripts are always shifted by 3 pixels. In IDXTextEdit the offset can be set freely.

The text in `TEHandle` is divided into runs of consecutive characters of the same style ([8], p. 261). These styles are saved in a `StyleRun` table in the `runs` field of the TextEdit style record. There is no way to save new styles (like super- or subscripts) to the `StyleRun` table. The solution was to implement a new style run table for the new styles and update this table every time the text is added to `TEHandle`. The new style table was implemented as a queue of type `IDXStyleQue`. Individual style runs are saved to the queue into elements of type `IDXStyleElem` (figure 5.5).

5.4.2 IDXTextEdit routines

IDXTextEdit replaces original TextEdit manager calls but it calls TextEdit manager whenever possible. IDXTextEdit has the same programming interface as the original TextEdit - the difference is that the names of the new routines begin with the string IDX. For example `TEStylNew` procedure is replaced by the `IDXStylNew` procedure.

The TextEdit manager provides so called hooks which can be used to implement extensions to TextEdit. Application programmers can install pointers (i.e. procedure pointers of type `procPtr`) to their own procedures into these hooks, so that instead of the default procedure the programmers own routine will be called. Procedure pointers are used in several places in the IDXTextEdit code. For example, the drawing of the super- and subscripts and pictures is handled by implementing a draw hook for TextEdit (routine `PascalDrawHook`).

In order to implement hypertext features to RTF windows the standard click loop routine had to be replaced by custom click loop routine. It sends a message `mouseDownInRTF` to HyperCard whenever the user clicks in RTF window. The `mouseDownInRTF` message is handled in the HyperTalk code in order to implement hypertext links from RTF windows.

In addition to the draw hook and click hook it was necessary to implement a width hook routine. It measures the length of new style runs, i.e. super- and subscripts and pictures drawn by the draw hook routine. The measurement is needed to map the mouse clicks to the actual character positions of the clicked text.

5.4.3 RTF file translation to `TEHandle`

When opening an RTF file it has to be translated to the formatted text. This is done by RTF Translator developed for RTF Converter and RTF Reader (Chapter 4). RTF Translator uses IDXTextEdit in order to produce super- and subscripts, mathematical formulas and pictures. RTF Translator opens an RTF file, reads it to the memory and translates RTF markup to the formatted text into `TEHandle`.

5.5 Authoring tools

In this section the authoring tools of HMLE are presented. The authoring tools are Node Tool for maintaining the node database and Link Tool for creating explicit links.

5.5.1 Node Tool

A node in HMLE is defined to be a file in the Macintosh file system. The node could be any file, but for mathematical purposes the possible node types are restricted to those presented in figure 5.6.

```
cSubChapter = 1;              { A subchapter. }
cDefinitionNode = 2;          { A definition node. }
cTheoremNode = 3;             { A theorem or lemma. }
cExampleNode = 4;             { An example node. }
cTableOfContentsNode = 5;     { Table of contents. }
cExerciseNode = 6;            { An exercise node. }
cMatlabNode = 7;              { A Matlab script. }
cCardNode = 8;                { A HyperCard stack. }
{ Educational activity nodes. }
cQuickTimeNode = 9;           { A QuickTime movie file. }
cMathematicaNode = 10;        { A Mathematica notebook. }
cMapleNode = 11;              { A Maple Worksheet. }
cEMakerNode = 12;             { An Exercise Maker document. }
cCMapNode = 13;               { A concept map node. }
```

Figure 5.6: Possible node types in HMLE.

A subchapter is the basic element in the HMLE. Subchapters contain the basic text for learning purposes. This text can be extended by definition nodes. They usually contain longer definitions of mathematical concepts than subchapters. Example nodes contain worked examples of the topics discussed in subchapters and possibly in definition nodes. Subchapters, definitions and examples are CNVR files so that they can be opened by RTF Reader and Exercise Maker. Other nodes are educational activity nodes. These nodes are opened by a tool program such as Matlab, Mathematica, QT Movie Player or HyperCard.

Nodes contain additional information that is saved into a special data structure `Node` (see figure 5.7). In the `Node` data structure the `ndFileType` and `ndCreator`

fields define the application that can open the node. Each node can have a set of keywords that describe the contents of the node (field `keywords`). Keywords are used to implement implicit linking and concept maps.

```
Node = record
    nodeName: str255;   { The name of the node. }
    nodeType: integer;  { The type of the node. }
    ndCreator: OSType;  { Creator signature. }
    ndFileType: OSType; { Finder file type. }
    reserved: integer;  { Reserved for future use. }
    keywords: str255;   { Keywords that describe the node. }
end;
```

Figure 5.7: Data structure `Node`.

Nodes form a node database for HMLE. It is maintained by Node Tool (figure 5.8). Node Tool displays the list of nodes in the database, and for a selected node it displays the node name, ID, type and keywords. By using Node Tool the author can add or delete nodes and change keywords or the type of the node.

The nodes in the node database can be accessed by the node name (function `GetNamedNode`), keywords or by the node ID (function `GetNode`). These procedures correspond to the accessor function in the Dexter Model.

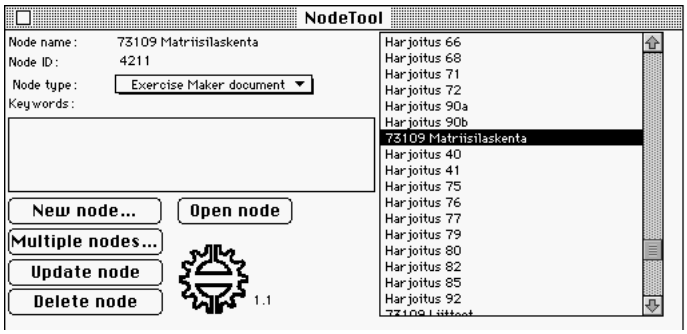

Figure 5.8: Node Tool.

5.5.2 Link Tool

There are two types of links in the HMLE. Explicit links are created so that the author explicitly defines the source and the destination of the link and saves the link to the database. This is done using Link Tool. Implicit links are generated by the computer automatically so that every mathematical concept in the text is considered to be a link into a predefined set of nodes containing all the definitions of mathematical concepts (nodes of type Definition). Implicit links are always reference links. Explicit links can be reference links or action links to some educational activity.

Explicit links

Explicit links are generated manually by the hypermedia author using Link Tool (figure 5.10). Explicit links are always string-to-lexia links. The data structure `Link` is defined in figure 5.9.

```
Link = record
    startCh: integer;       { Start character for the link. }
    endCh: integer;         { End character for the link. }
    targetNodeID: integer;  { Target node ID, if = 0 then... }
    targetCardID: longint;  { ...target is a card.}
    linkType: integer;      { The type of the link. }
    cardLayer:boolean;      { Field in card or background? }
    cardID:longint;         { Source card ID. }
    fieldID:longint;        { Source field ID. }
end;
```

Figure 5.9: Data structure `Link`

In the `Link` data structure the fields `startCh` and `endCh` define the character range from where the link is defined (source). The destination node is defined by the `targetNodeID` field. The `linkType` field defines the type of destination node.

The type of the destination node defines also the method as to how the target node should be opened. Explicit links are saved to the link database.

Figure 5.10: Explicit links are handled by Link Tool.

Implicit links

The Finnish language has many inflected forms of a word (i.e. word variants). The implicit link is a link from the inflected form of a concept to the destination node that defines the concept. The actual link is generated by the computer so that the basic form of the concept is generated by which the destination node can be recovered.

The problem of word variants can be partly solved using a *conflation algorithm* that is a computational procedure which reduces variants of a word to a single form [107]. There are also algorithms that can regognize and produce word-forms based on morphological analysis of the text [70]. Koskenniemi has designed such an algorithm based on lexicon and a set of rules that are derived from the syntax of a certain language [70]. The implementation of Koskenniemi's algorithm is about 2000 lines of Pascal code ([70], p.137). In addition, there is the need to design lexicon and the rules for the morphological analysis. As suggested by Koskenniemi, simplier methods still have practical use in some applications ([70], p.13). This is why a conflation algorithm based on right-hand truncation and dictionary lookup was designed for HMLE.

The conflation algorithm developed for HMLE works in situations where it is possible to make a list of the possible basic forms of mathematical concepts in alphabetical order i.e. a dictionary. The idea is to compare the inflected form of a concept (term) to the keys in the dictionary and to find the corresponding basic form, if it exists. In HMLE the dictionary is constructed from the keywords defined in each node. This type of program is not a complete way to search for the basic form of a word, but in this case it serves the needs of the hypertext.

The search problem: Search from the dictionary a key c that is similar to the search term s. The term s and a key c are similar, if after right-hand truncation of the term and the key the word bodies are equal.

The searching of the basic form of a concept has to be considerably fast. This is why the binary search algorithm was used to search for the key from the dictionary. When comparing the term and a keyword the last few characters are neglected from the key and the term. If the bodies of the term and the key match the algorithm has found a basic form for the term.

For example, the algorithm matches with the basic form of a word "matriisi" (matrix) and the inflected forms of "matriisit" (matrices), "matriisina" and "matriisin". However, it incorrectly matches the word "jälki" (trace) and the word "jälkeen" (after).

It is also possible to recognise basic forms of terms which contain more than one word by applying the conflation algorithm to all the words in the term. For example "Jordanin kanonisen muodon" (of Jordan canonical form) is matched to the "Jordanin kanoninen muoto" (Jordan canonical form).

5.6 Learning tools

In this section, learning tools of HMLE are presented. Learning tools are Exercise Maker for presenting mathematical hypermedia documents and interactive exercises and Concept Map tool for presenting concept maps.

5.6.1 Exercise Maker

Exercise Maker is the client application for HMLE Server. Exercise Maker displays CNVR documents and provides interface for interactive exercises (Paper 3), hinted exercises (Paper 4) and navigational aids for the user. Interactive exercises are generated by Mathematica [108]. Exercise Maker communicates with HMLE Server using Apple Events and with Mathematica using MathLink (Paper 3).

A hypertext node is opened in a window in Exercise Maker. The window includes a navigation strip below the window title bar (figure 5.11).

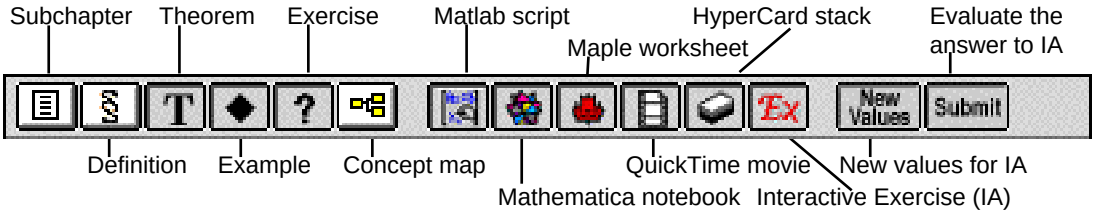

Figure 5.11: Navigation strip in Exercise Maker.

In the navigation strip there is a set of active and inactive buttons. Active buttons (white background) indicate available hypertext links from currently selected text. There are also buttons for generating new numerical values and for checking the answers for interactive exercises. Navigation strip corresponds to the Link Palette in HMLE Stack (Paper 2).

Anchors of the links are not emphasized in HMLE because mathematical hypermedia is typically *rich* hypertext i.e. mathematical text contains many mathematical concepts that are all linked to their definitions. In addition, emphasizing by using for example underlining may cause misunderstandings of mathematics, because in some notation, vectors are denoted as underlined lowcase letters. Instead, by selecting a piece of the text and checking active buttons in the navigation strip students discover if there are links from the selected text.

Exercise Maker includes a special type of a link, so called *folding link* (figure 5.12). Folding links are used to implement table of contents and hints in exercises. By default, the text of the exercise is opened and hints are hidden behind folded links. The student can open a folded link by clicking a small triangle in the folding.

Figure 5.12: Exercise Maker

5.6.2 Concept map tool

Concept maps have been useful tools in helping students learn how to learn. Concept map tool is discussed from the students point of view in Paper 2 and from the hypermedia authors point of view in Paper 4.

Educational books in mathematics contain a clear, hierarchical structure based on mathematical concepts, axioms, theorems, proofs, examples and exercises. The definitions of new concepts are often based on predefined concepts. Mathematical concepts can be given in a partial order that can be displayed as a graph. In mathematics, concept maps explain which concepts students should know when studying a new one.

In order to generate concept hierarchy from text material, the text should be structured. A simple structure can be implemented as a file hierarchy, where the definition of each concept is saved to a file. The file names can be considered as search keys. However, file names may often contain only a specific set of characters and their length may be restricted. A solution to the problem is that search keys

and corresponding file names are kept in a separate database. Another solution is to mark up the text with specific tags describing the structures of the text as discussed in Chapter 3.1.

Concept map tool presents the concept hierarchy as a tree. As the root of the tree there is a high level concept, for example "Singular Value Decomposition". The children of the root are concepts that are used to define the concept in the root. Similarily, each child is the root of their own subtree. The concept map shows which concepts the students should know before studying the concept in the root of the tree. The concept map clearly visualises the concept hierarchy and affects to the learning process by organizing the knowledge in the students mind. This is why the concept maps can be described as cognitive tools (Chapter 1).

5.7 Distribution of hypermedia

The availability of the hypermedia material greatly affects its usefulness [98]. In general, there are two possible ways to distribute hypermedia based learning environments. For stand-alone computers CD-ROMs are probably the only reasonable way to distribute software that uses more than 10 megabytes of disk space. For computers connected to networks Internet is another way to distribute the learning material. The drawbacks of Internet are that modem connections are still too slow to distribute for example large QuickTime movies. However, documents containing text and simple graphics can be distributed using modem connections (Paper 5). HMLE was made for distribution on CD-ROMs (Paper 2, Paper 6) and later was implemented also on the WWW (Paper 5).

5.8 Summary

In this section research problems presented in Chapter 1 and the proposed solutions are summarized. In general, they are presented in table 5.1.

The presentation of mathematical texts is often problematic in many current hypermedia systems (f. ex. HyperCard and WWW) as was shown in Chapter 2.

	Research problem	Solution in HMLE
RP 1.	Presentation of mathematical texts.	IDXTextEdit, RTF Reader and Exercise Maker.
RP 2.	Input, comparison and evaluation of mathematical expressions.	Exercise Maker by using Mathematica.
RP 3.	Conversion from ordinary text to hypertext.	Conversion of RTF files to CNVR files, RTF Converter.
RP 4.	Division of linear text to hypertext nodes.	Structure of mathematics, manual work.
RP 5.	Automatic link generation.	Implicit links in HMLE.
RP 6.	Integration of problem-solving tools and other cognitive tools into hypermedia environment.	Concept maps, interactive exercises, hinted exercises and animations.
RP 7.	Distribution of hypermedia material.	CD-ROM.

Table 5.1: Summary of solutions to research problems

In addition, current hypermedia environments do not support the use of cognitive tools as part of a learning environment whereas many CAL packages do not support hypertext.

The solution in HMLE is the special text edit package IDXTextEdit and software tools which use the package (RTF Reader, Exercise Maker). Input, comparison and evaluation of mathematical texts is partially solved in HMLE using Exercise Maker for the needs of Matrix Algebra course as presented in Paper 3. The solution is not complete but serves as the first step to user-friendy interface to the computer algebra system.

The solution to the conversion problem is the RTF Converter which converts RTF documents to hypertext. The solution to the division of mathematical documents to hypertext nodes is to divide the material manually into different types of nodes as presented in Paper 2. Automatic link generation and implicit links support the authoring of hypermedia material as discussed in Chapter 5.6.2 and in Papers 4 and

5.

Cognitive tools included in HMLE are Exercise Maker and interactive exercises discussed in Paper 3, concept maps discussed in Paper 4 and in Chapter 5.7, and hinted exercises discussed in paper 4. Papers 1 and 2 discuss the role of animations in a hypermedia based learning environment. The role of the simulations and the visualisations of mathematical models were discussed in Paper 1. The role of video clips in hypermedial learning environments are discussed in Paper 5.

Concept maps and interactive exercises are cleary cognitive tools because they require active and creative role from the user and thus stimulate the learning process in the students mind. They belong to the shaded octant in the cube of general framework of learning presented in the Chapter 1 (figure 1.2). Similarily, simulation tools affect the learning process positively and therefore also belong to the shaded octant in figure 1.2.

Chapter 6

Classroom Experiences

This chapter presents evaluation methods for computer-aided learning software and classroom experiences of mathematical hypermedia software presented in this thesis. The authoring of HMLE courses are discussed in Paper 4. Paper 1 presents four examples of learning with a mathematical simulation environment. In Papers 2 there is a discussion on different study styles in hypermedia based learning environments.

6.1 Evaluation of CAL software

There are several evaluation methods for computer-aided learning software. For example, the European Academic Software Award (EASA) organising committee has developed an extensive scheme for qualitative evaluation of the software participating in the 1994 EASA competition [39]. The following areas are evaluated in the EASA scheme: quality of the subject matter, appropriateness of the educational approach, the design of the interface, suitability of the software for use across Europe and innovative aspects of the software. Each area is evaluated in both verbally and numerically (1-5). The EASA evaluation method is intended to be used with external expert reviewers in the development phase of the CAL software.

Lifländer has developed a useful scheme for evaluating CAL software ([76], p. 42) that is based on cognitive learning theory. It has been used successfully in [77]. The scheme consists of 28 criteria grouped by contents, learning process, activation, openness, cooperativity, individuality, usability and security. Each criterion is eval-

uated using a numerical scale (1-5). For example, criteria for the content are: (i) Interesting, motivating, (ii) Visual, (iii) Many different views and (iv) Organising frameworks (orientation basis). Lifländer's method can be used for example when comparing different educational packages.

Laurillaird presents an evaluation method used in Technology-Enhanced Language Learning project [73] but it can be used in many other disciplines too. The evaluation is divided into formative and summative phases. Formative evaluation is used during the software design while summative evaluation is an extensive, large scale evaluation used to measure the success of the software. The formative phase enables designers to improve the software so that it meets the needs of learning the subject matter [73]. Summative evaluation answers questions like "How does the material embed in existing course" and "How well are students able to use what they have learned in other contexts" [73]. The methods of data collection are observation, interviews, questionnaires, pre- and post tests and feedback sheets [73]. The evaluation should be organised so that both students and teachers are involved in the process.

The evaluation of Matrix Algebra presented in the thesis belongs mainly to the formative phase in Laurillaird's method but it also includes elements from the summative evaluation. The following aspects were evaluated in the final questionnaire: the usage of central features of the software (hypertext links, Matlab exercises, hinted exercises, theory as hypertext) and their implications to the learning process, motivation, the learning process, the teacher's role and the relation of the hypermedia group to lectures and ordinary exercises. The complete questionnaire can be found in the appendix.

6.2 How the medium influences to what is taught

As discussed in Chapter 2, technology is only one method in the reform of calculus. In particular, by using technology in calculus courses it is possible to use complex, real world problems for introducing mathematical concepts without focusing on computational matters [37]. The availability of computer algebra system or numerical

software package makes it possible for the students to experiment with their ideas [71]. However, technology alone is not seen as the solution. Many improvements in the reform of calculus were derived from cognitive learning theory. In table 6.1 the cognitive learning model (new model) is compared to learning in ordinary classroom situation (old model) [94]. The new model emphasises individual learning by using computers that have access to large information resources (CD-ROM, networks) and team learning using collaborative tools and email. The apprenticeship and teacher as guide refers to the integration of expert guidance into the learning environment. If the students are connected to a computer network, the expert can be reached over the network. Otherwise the teacher in the computer laboratory should be the expert. The information in the new learning model may be fast-changing and easy to produce. As a conclusion, not only the medium affects what is taught but the learning theory affects the technology used.

Old model	New model	Technology implications
Classroom lectures	Individual exploration	Networked PCs with access to information
Passive absorption	Apprenticeship	Requires skills development and simulations
Individual work	Team learning	Benefits from collaborative tools and email
Omniscient teacher	Teacher as guide	Relies on access to experts over network
Stable content	Fast-changing content	Requires networks and publishing tools
Homogeneity	Diversity	Requires a variety of access tools and methods

Table 6.1: Cognitive learning model and its implications to technology.

6.3 Distributed Parameter Systems course

The TDP simulator discussed in Paper 1 has been used in the Distributed Parameter Systems course in the Department of Mathematics Department at TUT. It was used in laboratory demonstrations during the last month of the course. The Distributed Parameter Systems course was lectured in the spring of 1992, with approximately 20 students attending the course.

In laboratory demonstrations the students were given tasks such as: Design a controller for a given process and simulate the system. Finally, students were asked to analyse the results of the simulation by producing an animation.

During the course it was seen how important it was to visualize the solutions of the control systems. From the animation of the solution students could easily 'see' how their controller operated. For example, the concept robust was demonstrated by using two different controllers (robust and fast) on the same process. The change in value of a process variable did not affect the performance of the robust controller. However, the fast controller showed substantial changes in the form of the solution.

It was observed that the Macintosh version of the TDP-simulator was easier to use than the previous version in the mainframe VAX because of the graphical user interface and the open structure of the program. For the students the design of the control system was easy to understand when they actually saw the system model and could change the system parameters simply by clicking the mouse and entering the new values. In the terms of learning theory, the TDP simulator is a cognitive tool (Chapter 1).

6.4 Matrix Algebra

A Matrix Algebra course was lectured in autumn 1994 with the possibility to do the course by hypermedia based exercises. There were 6 hours of lectures per week and 4 hours of laboratory work using HMLE software. The course lasted for 7 weeks with about 150 students. The students were selected to the hypermedia group by taking 25 volunteers. The remainder did the course by doing homework and an

examination. All the students in hypermedia group were male. There were 8 female students out of 66 in the first examination of the Matrix Algebra course. In general, the percentage of female students in Matrix Algebra examinations has been between 5 and 15 in recent years.

The hypermedia group met in a computer classroom with 16 Macintosh IIsi computers which had HMLE for Matrix Algebra installed. The teachers computer also had a data projector for displaying the teachers material to the students. The computers were networked which made it possible to quickly share the teachers files with the students. There was also a laser printer for printing hypermedia material and Matlab solutions.

HMLE for Matrix Algebra contained exactly the same lecturing material as the printed version (about 140 printed pages). In addition, the hypermedia material contained about 60 pages of hinted exercises. The conversion of the material from text to hypertext took about 2 days i.e. 2 x 7 hours (all the material was available in electronic form).

At the beginning and end of the course the students were asked to fill a questionnaire. In the first questionnaire they were asked to evaluate their computer skills. In the second questionnaire they were asked to evaluate the course and their study styles compared to an ordinary mathematics course.

In the laboratory demonstrations, the students in the hypermedia group worked in small groups. They were asked to read the related theory using HMLE and solve a few hinted exercises using HMLE and Matlab. Every other week they also solved 3 to 5 exercises on paper and returned the solutions to the teacher. The grade of the course was determined based on the evaluation of the solutions.

During the first two weeks of the course 4 students left the hypermedia group mainly due to timetable reasons. 19 students passed the course, 13 with excellent grades. The 2 students who did not complete the course did not do enough evaluated exercises. They told the teacher that they wanted to come to the hypermedia group because it was good practice for the examination. They found hypermedia a motivating way to study for the examination.

6.4.1 Changes in teacher work

In ordinary exercises the teacher explains the exercise and writes the solution on the blackboard explaining each step in the solution process. Students copy the solution trying to learn at the same time. Finally, students may ask questions. Naturally, the preparation of hypermedia based hinted exercises requires considerably more time than the preparation of ordinary exercises.

Exercises in the hypermedia group were typical pen-and-paper exercises with added hints (Paper 4). Two types of hints were given. The first-level hint explained the question in more detail, the second-level hint revealed the structure and results needed in the proof. The students were asked to construct the entire proof step by step. Finally, complete solutions were given for some exercises. The extra time in preparation was used in designing the hints that would support learning i.e. the teacher had to think about the exercise more carefully from the point of view of the learning process.

Interactive exercises for the Matrix Algebra course were based on integer matrices as discussed in Paper 2. However, exercises are not restricted to include only integer matrices. The type of matrices can be freely selected by the exercise author.

In the learning situation the teacher's role changed from a dominant lecturer to a consultant, that monitored student's actions and proposed new ways of seeing the problem. In addition, the teacher gave guidance in the use of the software. The changes in the teacher's work reflect the ideas of the cognitive learning theory discussed in Chapter 1: the teacher does not deliver information but assists the students to assemble the knowledge. All but one student considered the teacher's role was suitable (not too active or passive).

6.4.2 Feedback from the course

At the beginning of the course the students were asked about their computing skills. Macintosh was unfamiliar to most of the students. Instead, many had used other computers with graphical user interface. Hypermedia was well known to almost all of the students, because they were familiar with WWW.

The students were asked why they chose to come to the hypermedia based learning group. Many regarded the ordinary lecturing as boring and wanted to test hypermedia for that reason. One student preferred not to go to the examination, but to do the course in some other way.

Students study styles can vary when studying with HMLE. In general, the text can be read lineary just like a book or more associatively using implicit links. Concept maps provide a more structured view to the subject matter (Paper 2). Studying can also be more problem-centered, as in the hypermedia group. In the learning situation the students formed small workgroups of two or three students. Together they discussed the given exercises and tried to find solutions. The communication between students was seen as an important factor in the learning process. In addition, the possibility to try out mathematical theorems with Matlab was seen as a way to "extended thinking". Almost all students realised that in an ordinary mathematics course they study the course only a couple of days before the examination. In this course they studied more intensively during the whole autumn. However, five students did not attend the lectures at all and 13 students said that they were absent from up to 4 lectures.

In the final questionnaire the students were asked to evaluate how they used the HMLE software. The results are presented in table 6.2. The most used features were hinted exercises and the possibility to use Matlab in conjunction with HMLE. Hypertext links also proved to be useful to this group. Minimum to moderate theory was used. This is because of the nature of the classroom situation where the students concentrated on exercises. If the material had been used at home the importance of theory pages would have increased.

6.5 Gifted mathematics students

A short course (20 h) on Matrix Algebra was given to gifted students age of 17-19. The students were mathematically oriented senior secondary school students studying in their last year school at Päivölä Institute. In addition to normal senior secondary school courses, their studies included first and second year undergraduate

	Links	Matlab	Hinted exercises	Theory as hypertext
Not at all	0	0	0	5.3
A little	42.1	0	0	42.1
Moderately	26.3	15.8	21	36.8
Much	21.1	73.7	63.2	15.8
Very much	10.5	10.5	15.8	0

Table 6.2: Use of HMLE features (%).

courses in mathematics and physics.

The Matrix Algebra course was available on two Macintosh computers and was used as an extensive material in the normal classroom situation. A few students preferred to study advanced material (matrix functions, Jordan canonical form) with the hypermedia, because they were already familiar with basic matrix algebra that was in the main group lectures.

Giving specialized exercises for students who are already familiar with the basic material should be seen as an important way of using CAL packages and hypermedia based learning environments.

6.6 Continuing education

The Matrix Algebra course was lectured in the spring of 1995 in the Pori Unit of TUT with the possibility to use the course over a network. Students were participating in continuing education for engineers. Most of the students had been working as engineers in companies and were very motivated in learning and improving their professional skills.

In the laboratory demonstrations (2 hrs per week) the Matrix Algebra WWW version [6] was used. It was also possible to use the material from home computers with a modem. Only a few students reported that they have a modem that is fast enough to be used with WWW but none actually used the material from their homes.

Certain topics were not lectured at all in the ordinary lectures but were left to be

studied with hypermedia material (such as vector norm). The students were asked to read the chapter on vector norms from WWW and after that do hinted exercises on the topic. In general, this was found to be a good way to study the subject. However, the network caused problems when 15 students tried to access the same material almost simultaneously.

6.7 Summary

Observations from the classroom experiences demonstrate that HMLE supports learning by providing a means to organize and present the information in alternative ways (i.e. linear text, implicit links from mathematical concepts to their definitions and concept maps) as discussed in Chapter 1. HMLE also supports problem solving by providing interactive exercises, hinted exercises and educational activity nodes, such as Matlab scripts (Paper 2). The examples from the Distributed Parameter Systems course showed how important it is from the point of view of understanding the theory to visualize the numerical solutions (Paper 1).

In the laboratory demonstrations of both the Distributed Parameter Systems course and the Matrix Algebra course, students were more active than in ordinary exercises. The student teams had a lot of discussion and cooperation together. The discussions between students are related to the development of the mental model of the learning topic (Chapter 1). Explaining their own ideas and how to solve the problem helps the students to reorganize their thoughts and thus learn better. The effect is not unique in hypermedia based learning but generally it can be found in all team work. However, the power of the hypermedia learning environment is that it supports the reorganization of the thoughts by organizing the material into the concept maps and providing hypertext links.

The changes observed in the learning of the hypermedia group are similar to what is presented in table 6.1. The learning in hypermedia group was clearly team learning where the teams explored the material individually. Apprenticeship was observed when the teams tried to solve exercises and team members discussed how they understand the problem. The teacher was a guide who was available in the

classroom, not over a network. The content in HMLE for Matrix Algebra material was easy to update, but in general, the content of the course was quite stable. A variety of access tools were available and were also used during the course.

HMLE for Matrix Algebra is evolving to contain more problems from engineering applications. Currently, such exercises are exercises on numerical methods and 3D-graphics (Paper 2). For example, control engineering and electronic engineering problems will be added to the course material. These kinds of exercises will serve as motivational factors, because students see in their early mathematics studies that they will need matrix algebra in their professional studies.

The preparation of hypermedia based hinted exercises required considerably more time than the preparation of ordinary exercises. However, most of the time were used to design exercises that support learning - the conversion of the material to the hypertext form was straightforward.

The most used features in HMLE for Matrix Algebra were hinted exercises and the possibility to use Matlab in conjunction with HMLE. It was possible for the students to quickly test their ideas i.e. validate their orientation basis of the learning topic (Chapter 1). The communication between students was also seen to be an important factor in the learning process.

Based on our experience there are a few ways to use hypermedia based learning environments. Hypermedia can be used in normal lecturing by replacing transparencies with hypermedia based presentation. The advantage is that in addition to actual text the teacher can quickly use the mathematical dictionary as a reference. Hypermedia learning environments can be used in laboratory exercises as a problem-solving tool. It is also possible to give specialized exercises to students who are already familiar with the basic material. As yet there are no experiences regarding the use of HMLE at home. However, it is believed that hypermedia based learning environments will also be useful when used at home to support self-studying.

Chapter 7

Conclusions

In this thesis hypermedia based learning environments for mathematics were defined and analysed. In addition, currently available CAL software for mathematics and hypermedia environments were characteristized and requirements for hypermedia based learning environments were proposed. The hypermedia environments that were reviewed, do not satisfy all the requirements that were set for a hypermedia based learning environment. The CAL software that was reviewed do not cover as much theory as is needed in a complete university course. On the one hand, many CAL systems do not support hypertext and the authoring of a CAL software is difficult and time-consuming. On the other hand, many hypertext systems as such do not support learning i.e. they do not include the possibility to use cognitive tools as part of a hypermedia environment.

The software package, called HMLE, developed in this thesis combines authoring and learning tools to a hypermedia based learning environment for mathematics. HMLE solves the problem of presenting mathematical texts in hypertext environments by introducing a special text edit package called IDXTextEdit and software tools (RTF Reader and Exercise Maker) which use the package.

The thesis also presented a solution to the conversion problem for converting mathematical documents created with Microsoft Word to hypertext. The conversion is semi-automatic: the author has to divide the text to nodes manually. However, the advantage of the conversion process is that it creates the structures for implicit

linking.

The problem of automatic link generation was discussed and as a solution the implicit links were presented. In addition to automatic linking, implicit links were used to design and implement concept maps, a cognitive tool that presents the hierarchy of mathematical concepts as a tree.

The authoring of mathematical hypermedia documents was also discussed. It was noted that authoring mathematical hypermedia using existing hypermedia systems is difficult. From this point of view, HMLE provides authoring tools that simplify the authoring process significantly, especially automatic link generation and implicit links support the authoring of hypermedia material. Still, the preparation of hypermedia based hinted exercises required considerably more time than the preparation of ordinary exercises. However, most of the time was used to design exercises that support learning - the conversion of the material to the hypertext form was straightforward.

The cognitive tools included in HMLE are Exercise Maker, interactive exercises, concept maps, hinted exercises and animations. They were seen as cognitive tools because they require an active and creative role from the user and thus stimulate the learning process.

The role of mathematical simulation environments within a hypermedia based learning environment was discussed and as an example the TDP simulator for distributed parameter systems was presented. HMLE does not yet contain simulation tools. This is not a problem, since HMLE allows the author to generate links to documents created with mathematical software packages, such as Mathematica, Maple and Matlab which can be used to implement simulations.

Observations from the classroom experiences suggest that HMLE supports learning by providing a means to organize and present the information in alternative ways (i.e. linear text, implicit links from mathematical concepts to their definitions and concept maps). Despite the classroom experiences collected so far, further study is needed to validate such an conclusion. This is partly due the fact that the monitoring of student actions is not yet implemented in HMLE.

Further research should concentrate on the question: Is there a special group of

students who get the best advantage from hypermedia based learning environments? From the educational point of view, another question arises: Do the students use different study styles in hypermedia based learning environments than when studying in the normal system?

Future research should also explore the implementation of hypermedia based learning environments on world-wide computer networks. WWW seems to be an attractive system for hypermedia based learning environments. However, authoring of mathematical documents for WWW is difficult since the authoring language HTML does not (as yet) support mathematical formulas. From this point of view, the use of structured mathematical documents in hypermedia environments should also be explored.

The communication between students was seen to be an important factor to the learning process. In the future, hypermedia learning environments could also include videoconferencing capabilities that would enable real-time communications between distance students and the teacher.

Bibliography

[1] Abbot, J., van Leeuwen, A., Roelofs, M., and Strotman, A. Objectives of OpenMath. Working paper, version 0.8.1, 1995. Available from http://www.rrz.uni-koeln.de/themen/Computeralgebra/OpenMath/om-obj-081.ps.

[2] The ACELA project. ACELA Project Factsheet. Available from CWI, Amsterdam, NL.

[3] Acrobat Exchange and Adobe Distiller 2.0, 1994.

[4] Ambron, S., and Hooper, K. *Learning with Interactive Multimedia.* Microsoft Press, Redmont, Washington, 1990.

[5] Andrews, K., Kappe, F., and Maurer, H. The Hyper-G network information system. *Information Processing and Management* (1994). Special issue: Selected Proceedings of The Workshop on Multimedia Systems.

[6] Antchev, K., Luhtalahti, M., Multisilta, J., Pohjolainen, S., and Suomela, K. *A WWW Learning Environment for Mathematics.* O'Reilly and Associates, Boston, USA, 1995.

[7] Apple Computer, Inc. *HyperCard User's Guide.* Cupertino, 1987.

[8] Apple Computer, Inc. *Inside Macintosh, Volume V.* Addison-Wesley, Reading, Massachusetts, 1988.

[9] Apple Computer, Inc. *Inside Macintosh, Volume VI.* Addison-Wesley, Reading, Massachusetts, 1991.

[10] Apple Computer, Inc. *HyperCard 2.1 Release Notes*, 1992.

[11] Apple Computer, Inc. ETO Essential tools and objects 15, 1994. The Development Tools CD Series.

[12] Asymetrix Corporation. *Using ToolBook*. Washington, 1989-1991.

[13] Balasubramanian, V. Hypermedia issues and applications: A state-of-the-art review. Working paper, CIS 776, New Jersey Institute of Technology, Newark, NJ, 1994. Available from ftp://ftp.cccc.njit.edu/pub/eies/hypertext_review.tar.Z.

[14] Bechtel, B. Inside Macintosh as hypertext. In *Hypertext: Concepts, Systems, and Applications* (Cambridge, UK, 1990), N. Streitz, Ed., Cambridge University Press.

[15] Beevers, C. E., Foster, M. G., McGuire, G. R., and Renshaw, J. H. Some problems of mathematical CAL. *Computers and Education 18*, 1-3 (1992), 119–125.

[16] Beilby, M. Mathwise: A computer-based learning environment. *Maths&Stats - the newsletter of the CTI Centre for Mathematics and Statistics 4*, 4 (1993), 2–5.

[17] Beilby, M. Making a feature of assesment. *Maths&Stats - the newsletter of the CTI Centre for Mathematics and Statistics 5*, 2 (1994), 6–10.

[18] Beilby, M., Bowman, A., and Bishop, P. *Maths&Stats Guide to Software for Teaching*, 2 ed. CTI Centre, University of Birmingham, Birmingham, UK, 1991.

[19] Bell, G. E., O'Connor, J. J., and Robertson, E. F. Mathematical MacTutor: A program for learning mathematics. In *Proceedings of Technology in Mathematics Teaching TMT 93* (Birmingham, UK, 1993), B. Jaworski, Ed., University of Birmingham, pp. 125–132.

[20] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. The World-Wide-Web. *Communications of the ACM 37*, 8 (1994).

[21] Bishop, P., Beilby, M., and Bowman, A. Computer based learning in mathematics and statistics. *Computers and Education 19*, 1/2 (1992), 131–143.

[22] Bogacki, P., Fife, E., and Husch, L., Eds. *Electronic Proceedings of the Seventh Annual International Conference on Technology in Collegiate Mathematics*. Mathematics Archives WWW Server, 1994. Available at http://archives.math.utk.edu/ICTCM/EP-7.html.

[23] Bryan, M. *SGML An Author's Guide to the Standard Generalized Markup Language*. Addison-Wesley, UK, 1988.

[24] Bush, V. As we may think. *Atlanic Monthly* (1945). Available from http://www.csi.uottawa.ca/ dduchier/misc/vbush/as-we-may-think.html.

[25] Campbell, B., and Goodman, J. M. HAM: General purpose hypertext abstract machine. *Communications of the ACM 31*, 7 (1988).

[26] Chignell, M. H., Nordhausen, B., Valdez, J. F., and Waterworth, J. A. The HEFTI model of text to hypertext conversion. *Hypermedia 3*, 3 (1991).

[27] Ciarlet, P. G. *Introduction to numerical linear algebra and optimisation*. Cambridge texts in applied mathematics. Cambridge University Press, Cambridge, 1989.

[28] Computers in Teaching Initiative. Annual report 1993-94 of the computers in teaching initiative. Tech. rep., CTISS Publications, University of Oxford, Oxford, 1994.

[29] Conklin, J. Hypertext: An introduction and survey. *IEEE Computer 20*, 9 (1987), 17–41.

[30] Cormen, T., Leiserson, C., and Rivest, R. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 1990.

[31] Cumming, G. Understanding and measuring the cognitive effects of CIT-mediated learning, 1995. Sixth World Conference on Computers in Education. Final report from professional group DG07. To appear in post conference report "IFIP Windows to the future".

[32] Daliz, W., M., G., Lugger, J., and Sperber, W. New perspectives of a distributed information system for mathematics. Technical report TR 94-8, Konrad-Zuse-Zentrum für Informationstechnik, Berlin-Wilmersdorf, Germany, 1994.

[33] Davis, B., Porta, H., and Uhl, J. *CALCULUS&Mathematica.* Addison-Wesley, Reading, Massachusetts, 1994.

[34] Davis, H., Hall, W., Heath, I., and Hill, G. Towards an integrated information environment with open hypermedia systems. In *Proceedings of the ACM Conference on Hypertext* (Milano, Italy, 1992), D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, Eds., ACM Press, pp. 181–190.

[35] Davis, R., Epp, S., Kenelly, J., Layton, K., Schoenfeld, A., Steen, L., Stein, S., Terry, S., and van der Vaart, H. Notes on teaching calculus. In *Toward a Lean and Lively Calculus*, R. Douglas, Ed., no. 6 in MAA Notes. The Mathematical Association of America, 1986.

[36] Del Corso, C. *Using MATLAB in the Classroom.* The MATLAB Curriculum Series. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[37] Douglas, R., Ed. *Toward a Lean and Lively Calculus.* No. 6 in MAA Notes. The Mathematical Association of America, 1986.

[38] Drakos, N. From text to hypertext: A post-hoc rationalisation of latex2html. First International Conference of the World-Wide-Web, 1994 at CERN, Geneva.

[39] European academic software award evaluation form, 1994. Unpublished.

[40] Edminister, J. A. *Electric Circuits - A MathCAD Electronic Book*. Shaum's Interactive Outline Series. MathSoft Inc., McGraw-Hill Inc., Cambridge, New York, 1995.

[41] Engeström, Y. *Perustietoa opetuksesta*. Valtion painatuskeskus, Helsinki, 1988. In Finnish.

[42] Entlich, R., Garson, L., Lesk, M., Normore, L., Olsen, J., and Weibel, S. Making a digital library: The chemistry online retrieval experiment. *Communications of the ACM 38*, 4 (1995), 54.

[43] Etter, D. M. *Engineering Problem Solving with MATLAB*. MATLAB Curriculum Series. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[44] Fox, E. A., Akscyn, R. M., Furuta, R. K., and Legget, J. J. Digital libraries introduction. *Communications of the ACM 38*, 4 (1995), 22–28.

[45] Frisse, M. From text to hypertext. *BYTE* (Oct. 1988).

[46] Gloor, P., Dynes, S., and Lee, I. *Animated Algorithms A Hypermedia Learning Environment for Introduction to Algorithms (CD-ROM)*. MIT Massachusetts Institute of Technology, Cambridge, Massachusetts, 1993.

[47] Goodman, D. *The Complete HyperCard 2.0 Handbook*. Bantam Books, New York, 1990.

[48] Gray, S., Kajler, N., and Wang, P. MP: A protocol for efficient exchange of mathematical expressions. In *ACM Proceedings of ISSAC T94* (Oxford, UK, 1994).

[49] Haan, B. J., Kahn, P., Riley, V. A., Coombs, J. H., and Meyrowitz, N. K. Iris hypermedia services. *Communications of the ACM 35*, 1 (1992), 36–51.

[50] Halaz, F., and Schwartz, M. The dexter hypertext reference. *Communications of the ACM 37*, 2 (1994), 30–39.

[51] Halaz F., S. M. *The Dexter Hypertext Reference Model.* No. 500-178 in NIST Special Publication. National Institute of Standards and Technology, Gaithersburg, MD, 1990. Proceedings of the Hypertext Workshop, January 16-18.1990.

[52] Hall, W. Ending the tyranny of the button. *IEEE MultiMedia 1*, 1 (1994).

[53] Hammond, N. Learning with hypertext: Problems, principles and prospects. In *Hypertext a psychological perspective* (New York, 1993), C. McKnight, A. Dillon, and J. Richardson, Eds., Series in Interactive Information Systems, Ellis Horwood, pp. 51–70.

[54] Harding, R., and Quinney, D. *MATHEMATICS Volume 2 (CD-ROM).* Anglia Polytechnic University, Renaissance Initiative, UK, 1992.

[55] Hill, D., and Zitarelli, D. *Linear Algebra Labs with MATLAB.* Macmillan Publishing Company, New York, 1994.

[56] Hill, G., and Hall, W. Extending the Microcosm model to a distributed environment. In *ECHT'94 Proceedings, ACM European Conference on Hypermedia Technology* (1994), pp. 32–40. Edinburgh, September, 1994.

[57] Huges, K. Entering the World-Wide-Web: A guide to cyberspace. Honolulu Community College, 1993. Available from: http://www.hcc.hawaii.edu/guide/www.guide.html.

[58] Johnston, V., Ed. *Educational Technology Abstracts*, vol. 9. Carfax Publishing Company, 1993.

[59] Jonassen, D. H. What are cognitive tools? In *Cognitive Tools for Learning*, P. A. M. Kommers, D. H. Jonassen, and T. J. Mayes, Eds. Springer-Verlag, Heidelburg, 1992, ch. 1, pp. 1–6.

[60] Jonassen, D. H., and Grabinger, R. S. Designing hypermedia for learning. In Jonassen and Mandl [61], ch. Problems and Issues in Designing Hypertext/Hypermedia for Learning, pp. 3–25.

[61] Jonassen, D. H., and Mandl, H., Eds. *Designing Hypermedia for Learning*, vol. F 67 of *NATO ASI Series*. Springer-Verlag, 1990.

[62] Kalaja, M., Lehtisalo, T., Hult, S., and Lassila, O. Implementing an authoring tool for educational software - the HyperReader experience. In *Proceedings of Nordic Conference on Computer Aided Higher Education* (Otaniemi, Finland, 1991), P. Uronen, Ed., Helsinki University of Technology, pp. 172–178.

[63] Kappe, F. M. *Aspects of a Modern Multi-Media Information System*. Dissertation for the award of the academic degree doctor of technical sciences, Graz University of Technology, Graz, Austria, 1991.

[64] Kent, P., Ramsden, P., and Wood, J. Transitional mathematics project. *CTI Maths & Stats Newsletter 5*, 1 (1994).

[65] Kent, P., Ramsden, P., and Wood, J. Mathematica for valuable and viable computer-based learning. In *Mathematics with Vision* (Southampton, UK, 1995), V. Keränen and P. Mitic., Eds., Computational Mechanics Publications, pp. 251–258. Proceedings of the First International Mathematica Symposium.

[66] Kivelä, S. K. Use of interactive mathematical programs as part of hypermedia learning systems. In *Proceedings of NORMA 94 Conference* (Lahti, Finland, 1994).

[67] Knowledge Revolution. Working model, 1994. Knowledge Revolution, San Mateo, USA.

[68] Knuth, D. E. *The TeXbook*. Addison-Wesley, Reading, Massachusetts, 1984.

[69] Kopponen, M., Kasurinen, V., and Linna, M. Experimentation of COSTOC-programs. In *Proceedings of Nordig Conference on Computer Aided Higher Education* (Otaniemi, Finland, 1991), P. Uronen, Ed., Helsinki University of Technology, pp. 246–253.

[70] Koskenniemi, K. *TWO-LEVEL MORPHOLOGY: A General Computational Model for Word-Form Recognition and Production*. PhD thesis, University of Helsinki, Helsinki, Finland, 1983.

[71] Lamagna, E., Hayden, M., and Johnson, C. The design of a user interface to a computer algebra system for introductory calculus. In *Proceedings of ISAAC-92* (1992), P. Wang, Ed., ACM Press.

[72] Lamport, L. *A Document Preparation System LaTeX User's Guide & Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[73] Laurillard, D. Evaluation methodology for a collaborative consortium. *The CTISS File*, 17 (1994).

[74] Lehtinen, J., Piipari, M., and Ropo, E. Tampereen yliopiston PLATO-kokeilu 1979-1980. Julkaisusarja A: Tutkimusraportti 18, Tampereen yliopiston kasvatustieteen laitos, Tampere, Finland, 1980. In Finnish.

[75] Lieslehto, J. *An Expert System for Multivariable Controller Design*. Publications 116, Tampere University of Technology, Tampere, Finland, 1993.

[76] Lifländer, V.-P. Tietokoneavusteisen opetuksen kehittäminen. Julkaisuja D 112, Helsingin kauppakorkeakoulu, Helsinki, Finland, 1989. In Finnish.

[77] Lifländer, V.-P. Project computer-aided learning at Helsinki university of technology 1991-1993. In *Proceedings of Hypermedia in Vaasa '93* (Vaasa, Finland, 1993), M. Linna and R. P., Eds., Vaasa Institute of Technology, pp. 209–212.

[78] Loimulahti, A. Hypermedia as motivation medium in mathematics learning. In *Proceedings of NORMA 94 Conference* (Lahti, Finland, 1994).

[79] Mannersalo, P. Selvitys matematiikan opetukseen soveltuvista tietokoneohjelmista. Reports C9, Helsinki University of Technology, Institute of Mathematics, Otaniemi, Finland, 1992. In Finnish.

[80] Marchionini, G., and Maurer, H. The roles of digital libraries in teaching and learning. *Communications of the ACM 38*, 4 (1995), 67–75.

[81] Marcus, M. *Matrices and Matlab: A Tutorial.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[82] McKenzie, R. Sub- and superscripting with TE. *MacTutor* (June 1989).

[83] Microsoft Corporation. *Rich Text Format Specification.* Redmond WA, Mar. 1993.

[84] Nielsen, J. *Hypertext & Hypermedia.* Academic Press, San Diego, 1990.

[85] Nievergelt, J. A pragmatic introduction to courseware design. *IEEE Computer 13*, 9 (1980), 7–21.

[86] Oinas-Kukkonen, H. Hypertext functionality in CASE environments: Preliminary findings. In *Proceedings of Hypertext in Vaasa '93* (Vaasa, Finland, 1993), M. Linna and P. Ruotsala, Eds., Vaasa Institute of Technology, pp. 227–234.

[87] Pam, A. Xanadu home page, 1994. Available from http://peg.pegasus.oz.au/xanadu/.

[88] Pantages, A. Control Data's education offering: "Plato would have enjoyed PLATO". *Datamation 22*, 5 (1976), 183–187.

[89] Pohjolainen, S., Multisilta, J., and Antchev, K. Examples of Matlab in the engineering education. Nato Advanced Research Workshop on Mathematical Modelling in Engineering Education, Izmir, Turkey.

[90] Pohjonen, J. Koulutusteknologia ja teknologia koulutuksessa (Educational technology and technology in education). In *Uusi teknologia koulutuksessa (New Technology in Education)*, K. Leiviskä and K. Schrey-Niemenmaa, Eds. University of Oulu, 1992. In Finnish.

[91] Rada, R. Converting a textbook to hypertext. *ACM Transactions on Information Systems 10*, 3 (1992), 294–315.

[92] Reader, W., and Hammond, N. Computer-based tools to support learning from hypertext: Concept mapping tools and beyond. *Computers and Education 22*, 1/2 (1994), 99–106.

[93] Reeves, C. Designing CAL to support learning: The case of multimedia in higher education. In *Proceedings of Nordic Conference on Computer Aided Higher Education* (Otaniemi, Finland, 1991), P. Uronen, Ed., Helsinki University of Technology.

[94] Reinhardt, A. New ways to learn. *Byte 20*, 3 (1995).

[95] Risku, P. Tietokonepohjainen matematiikan opiskeluympäristö. Tietotekniikan lisensiaattityö, Matematiikan laitos, Jyväskylän yliopisto, Jyväskylä, Finland, 1991. In Finnish.

[96] Russell, D., and Landow, G. Educational uses of hypermedia: From design to the classroom. Course material on ECHT 94, Edinburgh, Scotland, Sept. 1994.

[97] Schatz, B. Building the Interspace: The Illinois digital library project. *Communications of the ACM 38*, 4 (1995), 62–63.

[98] Siviter, D., and Brown, K. Hypercourseware. *Computers and Education 18*, (1-3) (1992), 163–170.

[99] Smith, A. HyperTeX, 1995. Available from http://xxx.lanl.gov/hypertex/.

[100] Strotmann, A., and Vorkoetter, S. Experiences with OpenMath. In *Proceedings of Third OpenMath Workshop* (Amsterdam, NL, 1995), RIACA.

[101] Stubenrauch, R. *Aspects of CAI and Hypermedia.* Dissertation for the award of the academic degree doctor of technical sciences, Graz University of Technology, Graz, Austria, 1992.

[102] Suutarinen, I. Hyperteksti ja hypermedia. Tietotekniikan pro gradu - tutkielma, Matematiikan laitos, Jyväskylän yliopisto, Jyväskylä, Finland, 1993. In Finnish.

[103] The MathWorks, Inc. *SIMULINK Dynamic System Simulation Software. Users' Guide.* The MathWorks, Inc., 1993.

[104] Turpin, S. J. TLTP institutional case studies. TLTP, Bristol, 1994.

[105] UK Mathematics Courseware Consortium. *Mathwise Courseware Design Specification.* Birminghan, UK, 1993.

[106] van Herwijnen, E. *Practical SGML*, 2 ed. Kluwer Academic Publishers, Boston, 1994.

[107] Willet, P. *Information Retrieval and Hypertext.* ACM European Conference on Hypermedia Technology, Edinburgh, UK, 1994.

[108] Wolfram, S. *Mathematica A System for Doing Mathematics by Computer.* Addison-Wesley, Redwood City, California, 1988.

[109] Wolfram Research Inc. *MathLink Reference Guide*, 1992. Technical Report.

Appendix: Final Quiestionnaire

MATRIX ALGEBRA I
Final Questionnaire

8.12.1994

Matrix Algebra I **Jari Multisilta**
Final questionnaire, Hypermedia group
8.12.1994

Please answer all the questions anonymously.

ISBN 951-722-561-X (printed version)
Reprinted from the original source to pdf 21.9.2004

1. Did you use hypertext links

not at all a little moderately much very much

2. Was the number of hypertext links

too little just enough too much

3. Did you use Matlab when solving the exercises

not at all a little moderately much very much

4. Was the number of Matlab exercises

too little just enough too much

5. Did you use hints

not at all a little moderately much very much

6. Was the number of hinted exercises

too little just enough too much

7. Did you read the theory in hypertext format

not at all a little moderately much very much

8. Was the amount of theory

too little just enough too much

ISBN 951-722-561-X (printed version)
Reprinted from the original source to pdf 21.9.2004

9. What was best in hypermedia group? (rank the top 3 items)

Hinted exercises
Lecture notes in a computer format
Concept maps
The possibility to use hypertext links
Possibility to use hypertext and Matlab at the same time
Possibility to work with a partner
Possibility to pass the course doing weekly exercises

Something else_____

10. What feature in the software helped you most to learn?

_____

_____

11. Why did you choose the hypermedia group?

_____

_____

12. Were your expectations about hypermedia learning material and the hypermedia group fullfilled ?

_____

_____

13. Compare your own working in the hypermedia group to the ordinary mathematics courses.

_____

_____

ISBN 951-722-561-X (printed version)
Reprinted from the original source to pdf 21.9.2004

14. Did you find the hypermedia group exercises

too theoretical	too practical	suitable
too long	too short	suitable

15. Was the teachers role in hypermedia group

too passive	too active	suitable

16. How often did you attend to the lectures of the course?
Every time
Missed 1-2 lectures
Missed 3-4 lectures
Missed more that 4 lectures
I did not attend the lectures at all.

17. How often did you go to the other exercise groups in addition to the hypermedia group?
Not at all
1-2 times
3-4 times
More than 4 times

18. What are your suggestions for improvement, criticism, praise, etc.

ISBN 951-722-561-X (printed version)
Reprinted from the original source to pdf 21.9.2004

Publications

Paper 1. Multisilta J., Pohjolainen S.: Hypermedia and Animation in Distributed Parameter Systems Education. *International Journal on Mathematial Education in Science and Technology.* 26(4): 599-618, 1995.

Paper 2. Pohjolainen S., Multisilta J., Antchev K.: Hypermedia Learning Environment for Mathematical Sciences. In Kajler N. (ed.): *Human Interaction in Symbolic Computation*; Texts and Monographs in Symbolic Computation, Springer-Verlag, 1998.

Paper 3. Antchev K., Multisilta J., Pohjolainen S.: Interactive Exercises on Matrix Algebra. *The Mathematica Journal*, Volume 7, Issue 3, 1999.

Paper 4. Multisilta J., Antchev K., Pohjolainen S.: Hypermedia for Mathematics: Authoring Courses with HMLE. *Proceedings of World Conference on Computers in Education*, WCCE'95 conference, Chapman & Hall, July, 1995.

Paper 5. Antchev K., Luhtalahti M., Multisilta J., Pohjolainen S., Suomela K.: A WWW Learning Environment for Mathematics. *World Wide Web Journal*, Issue One: Conference Preceedings, Fourth International World Wide Web Conference, Boston, Massachusetts, 11.-14.12.1995.

Paper 6. Pohjolainen S., Multisilta J., Antchev K.: Matrix Algebra with Hypermedia. *Education and Information Technologies*, IFIP TC 3 Journal, pp. 123-141, vol.1, no 2, 1996.

Tampereen teknillinen korkeakoulu
PL 527
33101 Tampere

Tampere University of Technology
P. O. B. 527
FIN-33101 Tampere Finland