1993

A hypertext based intelligent assistant for courseware preparation: a step toward authoring shells

Abdolhossein Sarrafzadeh

University of Wollongong

Recommended Citation

NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
A HYPERTEXT BASED INTELLIGENT ASSISTANT FOR COURSEWARE PREPARATION: A STEP TOWARD AUTHORING SHELLS

A thesis submitted in fulfilment of the requirements for the award of the degree

PH.D.

UNIVERSITY OF WOLLONGONG

BY

Abdolhossein SARRAFZADEH, B.Sc., M.Sc.

COMPUTER SCIENCE DEPARTMENT

1993
ACKNOWLEDGEMENTS

I would like to greatly thank my supervisor Prof. Fergus O'Brien for his valuable guidance and support throughout this work without which this thesis would not have been possible. I would also like to thank Dr. Ian Pirie my co-supervisor whose comments were always of help to me. Many thanks are due to Prof. Greg Doherty for reading and commenting on the most important parts of this thesis and for moral support I really appreciated receiving from him. Many thanks to Dr. Barry Harper for his supportive comments on my ideas at the initial and later stages of my work and for making it possible for me to use the Multimedia laboratory in the Faculty of Education. I would like to thank Mrs Jocelyn Booth for being very co-operative and helpful during the interviews I had to conduct for acquiring knowledge in the domain of geography. I would also like to thank those of her students at St Mary's College of Wollongong for participating in the testing of my prototype and the schools management for making it possible to do the test. I owe many thanks to all the support staff of the department and specially Philip Herring for being there whenever I needed their help. Last but not least, I would like to thank my wife for being patient during this study and doing parts of typing of this thesis.
Existing intelligent tutoring systems are mostly academic prototypes built for research purposes. The main reason for this is that they are time consuming and expensive to build. Another reason is that there are no tools that can be used for building such systems for teachers who usually are not programmers. In this study we have proposed authoring shells as a tool for building intelligent tutoring systems to be used as a goal for research in this area. As a step toward authoring shells, we have designed and implemented a hypertext based intelligent assistant (IACP) for developing courseware meant for use by teachers. To make IACP possible we developed a model for representing domain structure and a methodology for domain structure elicitation. The model we have developed is called the concept relationship model. The CR model is constructed through interviews with a domain and subject matter expert using the methodology developed in this study. A CR model was constructed, trialled and its validity demonstrated. The CR model can be used for automatic hypertext linking. This makes generation of hypertexts, both for the purpose of courseware preparation and self exploratory learning, possible. We have also introduced the notion of intelligent links which turn the hypertext into a semi-guided learning environment more suitable for learning. Based on the CR model, student modelling can be done. IACP generates a student modelling component with every collection of course material it retrieves, structures and links together. The CR model has the potential to be used as a hypermedia tool. Although we have developed the CR model for the purpose of courseware development it can be used for general hypertext generation. It solves the problem of disorientation by providing a higher layer of links above the document level. Teachers can use IACP for preparing course material. Students can explore the material structured by the system for the purpose of learning. An expert's knowledge of the domain structure is input to IACP by a knowledge engineer.
I hereby declare that I am the sole author of this thesis. I also declare that the material presented within is my own work, except where duly acknowledged, and that I am not aware of any similar work either prior to this thesis or currently being pursued.

Abdolhossein SARRAFZADEH
This thesis is submitted to the University of Wollongong, and has not been submitted for a higher degree to any other university or institution.

Abdolhossein SARRAFZADEH
# TABLE OF CONTENTS

1. INTRODUCTION ........................................................................................................ 1
   1.1 GENERAL .............................................................................................................. 1
       1.1.2 Authoring Shells ......................................................................................... 2
   1.2 Statement of the Problem .................................................................................. 5
   1.3 An Intelligent Assistant for Courseware Preparation ...................................... 8
       1.3.1. The Concept Relationship Model ............................................................... 9
       1.3.2 Disorientation Free Automatic Hypertext Creation ................................. 14
       1.3.3. Architecture of the System ..................................................................... 18
   1.4. Objectives and Scope of the Study ................................................................ 19

2. Artificial Intelligence in Education ......................................................................... 23
   2.1 Computers in Education ................................................................................... 23
   2.2 Artificial Intelligence, Expert Systems, and Instruction ............................... 27
   2.3 History of CAI Systems .................................................................................... 28
       2.3.1 Some Significant Systems in the History of CAI .................................... 30
   2.4 Intelligent Computer Aided Instruction (ICAI) ............................................. 35
       2.4.1 Problems of ICAI .................................................................................... 37
       2.4.2 CAI Versus ICAI .................................................................................... 39
       2.4.3 Related Work ........................................................................................... 42
       2.4.4 ICAI Systems Architecture ................................................................. 44
           2.4.4.1 General ............................................................................................... 44
           2.4.4.2 The Functional Approach ............................................................... 45
               2.4.4.2.1 The Tutoring Module ................................................................. 45
               2.4.4.2.2 The Knowledge Base ............................................................... 46
               2.4.4.2.3 The Student Modelling Module .............................................. 47
               2.4.4.2.4 The User Interface ................................................................. 49
               2.4.4.2.5 The Bite Sized Approach ....................................................... 50

3. Representing Domain structure Using the Concept Relationship Model ........... 51
   3.1 Representing the Domain ................................................................................ 51
       3.1.1 Knowledge Representation for ITSs ......................................................... 52
       3.1.2 The need for Multiple Formalisms for Representing Domain knowledge 54
   3.2 Relevant Cognitive Psychology Research ...................................................... 54
   3.3 Concept Mapping ............................................................................................ 59
       3.3.1 A Procedure for Constructing Concept Maps ........................................... 65
   3.4 Semantic Networks .......................................................................................... 67
   3.5 The Concept Relationship Model ................................................................ 68
       3.5.1 What is a Concept? .................................................................................. 70
LIST OF FIGURES

Figure 1.1 Relationship Among Courseware Development Tools and Level of Code and Contents Separation ....................................... 2
Figure 1.2. An Authoring Shell Represented as a Pyramid .......... 4
Figure 1.3. Useful Material Downloaded to Disk .................. 10
Figure 1.4 The Intelligent Assistant Simplified ..................... 10
Figure 1.5 A Graphical Representation of the Concept Relationship Network ................................................................. 11
Figure 1.6 Hypertext Version of Pages of Wollongong University Calendar ................................................................. 13
Figure 1.7 Lost in the Hyperspace ....................................... 14
Figure 1.8 (a) Multi Layered Hypertext Linking .................... 15
Figure 1.8 (b) Multi Layered Hypertext Linking .................... 16
Figure 1.9. Architecture of the Intelligent Assistant ............... 17
Figure 2.1 Common Components of ICAI Systems and Their Relationship ................................................................. 46
Figure 3.1 Multiple Types of Representation and Multiple Formalisms Needed in ITSs ....................................................... 55
Figure 3.2 Classification of Concept Maps ............... 63
Figure 3.3 A Portion of a CR Network ............................. 71
Figure 3.4 A Multi-layered CR Network ........................... 76
Figure 3.5 Summary of the Node of Development from a Higher Level Network ....................................................... 77
Figure 3.6 The Life Cycle of Knowledge Acquisition for Expert Systems ................................................................. 86
Figure 3.7 Structure Elicitation Life Cycle ......................... 90
Figure 4.1 Links in Hypertext ........................................... 102
Figure 4.2 Bush's Idea of a Network of Linked Documents (Memex) ................................................................. 104
Figure 4.3 A Typical Screen from NoteCards Showing FileBox and Text Cards ....................................................... 117
Figure 4.4 An Intermedia Screen (Bio106: Cell Biology in Context) ................................................................. 120
Figure 4.5 A KMS Frame ................................................. 123
Figure 4.6 The Hypertext Subsystem ................................ 127
Figure 4.7. An Example of a CR Card ................................ 128
Figure 4.8 Hypertext Document Stacks and their relationship with the Concept Relationship Network .. 129
Figure 4.9 An Example of a Text Card ................................ 130
Figure 4.10 An Example of an External navigation Card ....... 133
Figure 5.1 Examples of Planning Rules Using Assertions About Student's Knowledge

Figure 5.2 An Example of a Probabilistic Grammar and a Sample Derivation

Figure 5.4 Student's Knowledge vs. System's Knowledge

Figure 5.5 Relationships Among the Components of GUIDON's Student Model

Figure 5.6 Part of the Genetic Graph Used in WUMPUS

Figure 5.7 Student Modelling Module Using Perturbation Modelling Techniques

Figure 5.8 Construction of the Differential Model Used in WEST

Figure 5.9 An Example of a Procedural Network for Subtraction

Figure 5.10 A Question Generated From Text by IACP

Figure 5.11 A List of Concepts Displayed for the Student to Select From

Figure 5.12 Students Asked to Identify Relationship

Figure 5.13 Post and Prerequisite Concepts Represented as a Graph

Figure 5.14 A Lesson Hierarchy Obtained from the Pre/Postrequisite Graph

Figure 6.1 Files and Stacks Needed for Starting IACP

Figure 6.2 IACP's Welcome Screen
Table 2.1 Summary of Some ICAI Systems

Table 6.1 Levels and Ranges of Relationships

Table 6.2 Summary of Student Models Constructed by IACP
1. INTRODUCTION

1.1 GENERAL

Early Computer Assisted Instruction (CAI) systems were constructed using general purpose programming languages. Authoring languages, authoring systems and other tools some of which employed AI techniques, were later developed for easing the process of courseware creation. As Figure 1.1 suggests, the trend has been towards separation of code and contents, which in turn brings about more flexibility.

Starting in the early 70's, researchers in AI have been applying the results of their research to CAI [Carbonell, 70]. Intelligent Computer Assisted Instruction (ICAI), also referred to as Intelligent Tutoring Systems (ITS), has become a subfield of AI.

Several areas of AI research impinge on ITS. For example, expert system shells are useful expert system building tools which allow a user not familiar with computer programming to construct expert systems. This is done by formalising the domain knowledge in the language of the shell, which is usually close to natural language. It is possible to think of a similar tool, which we call an authoring shell, for constructing an ITS.
Figure 1.1 Relationship Among Courseware Development Tools and Level of Code and Contents Separation

1.1.2 Authoring Shells

The area of Intelligent Tutoring Systems is still at the research stage [Elson-Cook and O'Malley, 90; Briggs and Brough, 90]. There have been only a few efforts based on the idea of using AI techniques and methods to construct general purpose tools for building intelligent tutoring systems [Sleeman, 86; Nicolson, Scott, Gardner, 88]. Furthermore, many of AI methods and techniques are still in their infancy and some are not at a stage to be used in computer based instructional systems. For instance, automated knowledge acquisition is not at a stage to be applied comfortably to construction of instructional shells, but it has the potential to be. Even though learning tutors such as the quadratic tutor [Sleeman and Brown, 82] have been constructed, it is still too early to have a general purpose shell which can learn a better solution method from a student and add it to its knowledge.
We have used the term "Authoring Shells" to refer to an ideal general purpose tool for constructing intelligent tutoring systems. One may think of an authoring shell as a pyramid with a changeable tip as shown in Figure 1.2. The tip of the pyramid will contain domain specific layers of domain knowledge, Text, graphics, Video, etc. The bottom of the pyramid will contain powerful tools such as automatic knowledge acquisition tools, natural language processing tools, the intelligent assistant constructed in this study, and other tools for the purpose of constructing intelligent tutoring systems and mechanisms for manipulating the various types of knowledge contained in the tip of the pyramid. Although such a concept is beyond current technology this study contributes to the idea of authoring shells and the tool constructed is a component of the ideal authoring shell.

Some AI methods and techniques can be applied to the construction of intelligent tools and assistants which can make the construction of instructional shells easier. Attempts with the aim of bridging the gap between conventional CAI and ICAI [Elson-Cook and O'Malley, 90] have been made. However, the existing intelligent tutoring systems (see Chapter 2) are mainly academic prototypes and are limited. Domains as limited as subtraction and addition have been used to experiment with.

In recent years, optical storage devices with very large storage capacity have emerged. Educational research has been taking advantage of this technology. CD ROMs which are storage
devices of high capacity are very attractive devices for mass storage in applications where there isn't a rapid change in the data. With the capacity provided by CD ROMs, several books or encyclopedias can be stored on a single disk. To make use of such masses of information, intelligent tools are needed. A special tool for this purpose and as a step towards authoring shells, has been constructed in this study. The user of such tools may be a teacher or a student using the tool to explore a subject or to prepare a course on a certain subject.

CD ROM disks are high density read-only storage devices, very suitable for applications (i.e. educational applications) in which
data does not change rapidly. They are not, however, suitable for transaction processing type applications. A CD ROM can hold large volumes of information three orders of magnitude larger than floppy disks with higher reliability. This capacity is equivalent to 15000 pages of text [Laub, 86].

Knowledge from a certain domain, in the form of text or graphics(etc.), can be put on CD ROM and into the hands of the user who may very well be a teacher. To make use of a CD ROM containing domain knowledge for educational purposes, it is necessary to manage such a bulk of information in an intelligent manner and extract what is necessary for a course. In other words a teacher needs an intelligent assistant to be able to see what is useful and related to the subject s/he is trying to construct a lesson on, using the CD ROM. An active student could learn by exploring a subject using the same tool. This involves issues related to information retrieval, artificial intelligence and domain model construction. To give the user the ability to navigate through the retrieved material, hypertext techniques were used, and a model for automatic hypertext linking has been developed in this work.

1.2 Statement of the Problem

Producing courseware is a time consuming task. Most of the time is spent on putting the material together and organising it in an effective sequence. In intelligent tutoring systems, student models are used for sequencing of the material. The student model helps the intelligent tutoring system individualise the material to
suit each student, depending on their level of knowledge. Misconceptions of the learner are sometimes taken into account. This is all in the context of guided learning, where the learner is guided through the learning material. However, this guidance is based on the student's understanding of the domain, and the system's knowledge of this understanding, which is initially acquired and is maintained throughout the interactions with the learner. Existing intelligent tutoring systems are still rather limited and the time spent on producing them is considerable. Automating the task of preparation of material and construction of an initial student model would be a desirable step toward reducing the time spent on producing such systems. This will make rapid production of courseware by the end user possible. With the current mass storage technology, huge amounts of information on a topic can be placed on a single CD ROM. A tool which can put related material on a topic together using stored and updated material put in the hands of teachers, who must eventually be the real producers of courseware, will make tutoring systems rapidly produced and thus more generally usable.

The automatic organisation of material has its problems. Although the risk can be reduced, and has been in this work using innovative methods, the material retrieved may not be all of what the user expects. This is known as recall in the context of information retrieval. Further, some of the material may be unrelated (precision). The choice can be resolved by aiming at high recall, which will usually mean low precision.
Hypertext is a technology with several applications, including its applications in learning and educational design. Hypertext is used in what may be called discovery learning where the learner explores the learning material by navigating through the material which could, in some cases, be multimedia. This study proposes techniques for making hypertext usable as a guided learning medium. Although learning applications of hypertext are considered, hypertext is used as a tool for the teacher to navigate through the material put together for the course/lesson of his/her choice, in an attempt to exclude unrelated material and make comments or include additional material. The same technology is used, together with other methods, for constructing an overlay model of the learner.

Using hypertext as a technology to improve on the problems of courseware production, with the current state of the art hypertext technology, would only impose an additional burden with respect to time on the producer of courseware. Hypertext linking is done manually and only little success has been gained in automating the generation of hypertext links. As Shneiderman [Shneiderman, 89] a well known researcher in the field of hypertext, states, automatic hypertext generation is a difficult task. The hardest part is recognition of links not explicit in the structure of documents. Automatic hypertext generation is a contribution of this study that not only makes hypertext usable for the purpose of courseware preparation, but makes automatic hypertext creation, in general, possible.
In summary, a reduction in time spent organising material for a course and ease in the organisation is an advance that, if made, will make construction of courseware more rapidly and easily possible. This allows the teachers, who are the main users of such systems, to construct courseware quickly and easily. This in turn increases the usefulness of computers as a tool in education. Another task requiring time is construction of student models. Automating the construction of an initial student model is another problem dealt with in this study.

1.3 An Intelligent Assistant for Courseware Preparation

What is proposed here is to take advantage of information on slow CD ROMs, or, for that matter, any mass storage medium, and selectively extract material related to a certain subject (and perhaps download it to a faster magnetic medium) as shown in Figure 1.3. This information can later be sequenced in accord with a teaching strategy using intelligent or conventional tutoring tools.

Figure 1.4 depicts this intelligent assistant (IACP). As shown in Figure 1.4, three types of users, namely, expert(s), teacher(s), and students, interact with the intelligent assistant. A model of the domain is constructed through interactions with the expert. The model is basically a network of concepts linked through their relationships. This network allows specific information concerning each concept (i.e. keywords defining it, difficulty level, hierarchy, etc.) to be stored. The model is constructed and corrected through the interactions with an expert.
An active student being asked to explore a subject "X", can use the tool for this purpose. A teacher uses the assistant to construct a course on a desired subject. A teacher using the tool can alter the conceptual model. Such an intelligent assistant is necessary at this stage, and has several advantages. Some of these advantages are [Sarrafzadeh, 93]:

1. Reduction in development time.
2. Students can use it to explore a subject within the domain.
3. Teachers can use it to set up a lesson (a sequence of concepts to be presented).
4. The conceptual model can be used as a basis for initialising the student models if the target authoring environment provides for such a facility.
5. Useful material is extracted from CD ROMs (Figure 1.3).
6. The conceptual model gives a general view of the contents.
7. The model can be altered by the user (teachers).

1.3.1. The Concept Relationship Model

What we need in this tool is an abstraction of the huge amount of knowledge we are dealing with on the mass storage device. Similar to a concept relationship model, a conceptual model is a model of the world constructed for a specific purpose. According to Sowa [Sowa, 84], this is a limitation of such models which makes them imperfect models. However, they fit our
knowledge representation needs, and what is considered in general to be a limitation by Sowa is a merit in this context.

The model which is to be used here is presented to the user as a network of concepts and their relationships, if any. The nodes represent concepts and relationships among them. A concept
As shown in Figure 1.5, the model consists of concept nodes and relationship nodes. Each of these nodes will have a corresponding node descriptor. A node descriptor for a concept node will have the following fields:
. a difficulty level
. a set of keywords
. a set of weights associated with the keywords
. a hierarchy level
. a set of synonyms
. domain specific knowledge
. related concepts
. other information

The difficulty level of a concept, assigned by the expert, is a relative value between 1 and 10 which determines how difficult a concept is, compared to the other concepts. This is particularly important for initialising the student model.

A dictionary of concepts related to the domain of interest defines each concept in terms of some keywords. A dictionary of synonyms is also used in conjunction with the dictionary of concepts. When the network of concepts is constructed, the keywords are retrieved from the dictionary. Each keyword is given an initial weight of $1/n$, giving them equal weights. The weights can be modified by the end user of the system.

The hierarchy level is assigned to each concept by the expert. This is going to structure the network into a hierarchy which can be used in constructing the lessons.
A relationship node descriptor for a relation node will include the following fields:

- strength
- keywords in common between the participating concepts
- other information

The degree to which two concepts are related is a numerical value corresponding to various strength levels, namely, very weakly related, weakly related, related, strongly related and very strongly related. It shows how strongly two concepts are related. A user may choose to only deal with strong relations. A relation may also be direct or indirect, which is implicit in that if there are direct

![Diagram of relationships between courses]

Figure 1.6 Hypertext Version of Pages of Wollongong University Calendar
Figure 1.7 Lost in the Hyperspace

links, then the relationship is direct, otherwise it is indirect. This will be made clearer when hypertext issues are explained in Chapter 4.

Related documents may have keywords in common. These will be a component of the descriptor of the relation between the concepts.

1.3.2 Disorientation Free Automatic Hypertext Creation

Hypertext is a collection of text with paths through related material. It has been used for the purpose of learning in education and has many other applications. Any text can be converted into a
hypertext once these paths are stablished. For example, a university calendar converted into a hypertext may have the pages introducing the computer science department linked to those of mathematics based on prerequisites of computer science courses.

![Diagram of multi-layered hypertext linking]

**Figure 1.8 (a) Multi Layered Hypertext Linking**

One of the main problems in hypertext systems is the problem of disorientation. Users get lost navigating through the space of links among the nodes. A user navigating through the University of Wollongong calendar may get lost in the space of linked text when exploring a link from the word mathematics which is connected to a religious text written by Alan Turing the famous mathematician (Figure 1.7). The Concept Relationship (CR) Model was used to construct a hypertext system with very little risk of disorientation. The CR Model acts as a higher level
intelligent layer over the hypertext document base. The users are always able to see where they are, by going up one level into the CR network. Figure 1.8 (a,b) illustrates this multilevel linking.
Figure 1.9. Architecture of the Intelligent Assistant

made possible through the CR Model. The CR Model is proposed in this study as a means of automatic generation of hypertexts.
1.3.3. Architecture of the System

The components of the intelligent assistant are shown in Figure 1.9 which is a more detailed view of Figure 1.4. As shown in the figure, the following components are present:

- a conceptual model
- a knowledge base
- a retrieval component
- a student model generator
- a hypertext component
- thesauruses (concepts and synonyms)
- an inference engine
- a User Interface
- a hyperdocument base
- a user

The knowledge base may contain different types knowledge which will be used by the inference engine. A domain specific thesaurus, both for the concepts and the keywords, is used. The concept relationship model contains a model of the domain in terms of concepts and relationships, which will be used by the retrieval component to retrieve relevant documents, and by the hypertext component to both generate the hypertext and to link it together. The Hypertext component will consist of linked documents and the mechanism for generating the links. As mentioned earlier, the links are multilevel. The user interface is responsible for the interactions between the user and the system. Each of these
components will be explained in more detail in the coming chapters.

1.4. Objectives and Scope of the Study

Problems addressed in this study were stated in a previous section. It is impossible to draw a solid line between the problems and objectives of the study, where necessary objectives were mentioned without elaboration.

An authoring shell is a tool introduced and thought to be necessary in this study, which can be used for constructing effective intelligent tutoring systems. An authoring shell with the specifications explained earlier is a task beyond the scope of this study with the current state of the art technology. However work done throughout this research may be considered as steps towards such authoring shells. Contributions are made to both courseware construction and hypertext creation. Chapter 1 is an introduction to the work done in this dissertation.

As stated earlier, constructing tutoring systems is a time consuming task. This and other research is aimed at simplifying this task. The objective of this study is to construct a tool which is a useful and intelligent assistant in extracting related material from a large store of information (content), which may be a CD ROM, and prepare it for use in a conventional or intelligent tutoring system. A system has been constructed for extracting and automatically linking relevant material for a lesson on a specific subject and constructing an initial overlay student model for a
teacher preparing a lesson using an intelligent tutoring system building tool. It may also be used by a student exploring a subject. The system constructs a model of the domain using a proposed concept relationship model. This model is constructed with the help of a domain expert. The proposed model, namely the CR Model explained in the next chapter, can be very useful in other applications.

The intelligent assistant may reside on the CD ROM where the information is stored. The final product may be used by end users to prepare lessons or to explore a specific subject in the domain. The idea may be extended to cover multiple domains which will bring other issues to light, such as concepts having totally different meanings in different disciplines, and other problems which will have to be dealt with.

To make hypertext more suited to both educational and general applications, the notion of intelligent document level linking, based on the concept relationship model, has been proposed. The intelligent assistant constructed in this study is, however, built with the aim of courseware preparation and with the teacher in mind. Although learners can use it and benefit from it, because a student model is constructed for guided learning, this sort of linking does not seem to be necessary. However, it does lead to a useful method for making hypertext more suitable for education. Using the hierarchy levels introduced in the CR model, navigation can be restricted so that it takes place with respect to the levels of the hierarchy. Links have prerequisite nodes in this context. Although there may be a link between two nodes at
document level, the user will have to visit prerequisite nodes before visiting their desired node. This can be improved using the initial model of the learner also constructed by the system. The prerequisite and hierarchy information is contained or is produced using the CR model. This may be a deviation from self exploratory learning using hypertext. It is however in no way a disadvantage. The learner eventually reaches the node s/he has selected to explore.

Generating hypertexts is itself a time consuming task with the current technology. Hypertexts are constructed by manually linking the plain text being converted into a hypertext. Using the CR model a hypertext is generated automatically. A customised network of a domain can be created and almost any text in that domain may be converted into hypertext. The networks are created by experts in the domain of interest. Construction of hypertexts can be made possible with little effort if such networks are made commercially available. The user of a hypertext is usually only interested in a portion of the text. Through the CR model, only that portion of the text related to the concepts of interest to the user is converted into hypertext.

Hypertext suffers from the problem of disorientation. CR networks provide for a medium of disorientation-free navigation through the text space. CR networks operate at a higher level layer above the hypertext. Hypertext issues are presented in Chapter 4.

Although information retrieval is not the focus of attention, it plays an important role in this work. In fact, the CR networks
play the role of a layer above the keyword and retrieval functions layer. It is common practice to analyse each piece of text manually and associate keywords with it. Works of Jones [Jones, 91] and most library systems are based on this principle. Natural language processing view of text retrieval is not in agreement with any such work without an understanding of the text. The CR model approach takes the middle path between the two. This approach bases the retrieval on an understanding of the domain rather than the text itself. The CR network does not have the inflexibility of having to associate keywords to each new piece of text. Adding new material or changing the text base does not require a replacement of the CR network. The network need not change as new material is added. It is, however, maintained as the field changes. This approach does not require such an extensive effort, nor suffer from the limitations of the text understanding approach. This again is a by-product of this research and is not a main concern.

Concepts are connected through relationships. Each relationship is given a degree of strength by the expert. Once text is retrieved, by comparing the retrieved text, the degree of relationship can be calculated. This is similar to a method of learning known in computer science as parameter learning. Chapter 6 explains text retrieval and learning possible through the CR model.

Using the CR model, an initial overlay model of the learner is constructed. Student modelling using the CR model is discussed in Chapter 5.
Acquisition of knowledge from the domain expert for constructing a CR network is the most critical task and in fact the bottle-neck in this context. The CR model, problems of knowledge acquisition, and methods for constructing CR networks for courseware preparation, are discussed in Chapter 3.

Knowledge acquisition can be automated. This is specially true with knowledge acquisition for constructing CR networks. This is, however, beyond the scope of this study. Knowledge was acquired from the expert and hand-crafted into a CR network which is used in this study.

The field of geography was chosen and used to illustrate the applicability of the ideas introduced. The system can be extended to treat a variety of other domains. The intelligent assistant constructed in this study and its application to courseware preparation in the domain of geography is presented in Chapter 6.

Literature in the areas involved was studied and reviewed and some references were used. Reviews of the literature are included where necessary, and references are given at the end of this dissertation. Appendices contain a user guide, amongst other material.

2. Artificial Intelligence in Education

2.1 Computers in Education
A Computer Based Instruction System is a program, or a series of programs, that provides a medium for application of computers to Instruction. This area of application of computers is usually known as Computer Assisted Instruction. There are, however, a variety of other names which are used to identify this field of study. We will mention some of these later in this section.

The software which is developed for instruction is usually called courseware and there are two approaches to courseware development. The first approach does not incorporate any form of artificial intelligence programming. Such systems may, however, include very sophisticated models and operations in their subject domain. Systems based on this conventional approach do not actually comprehend anything about the student, material, or pedagogical approach. The second approach, or the artificial intelligence approach, to computer assisted instruction applies the results of research from various fields of artificial intelligence and cognitive science to the development of intelligent computer assisted instruction systems. One such area of AI research, which seems to have been appealing to ICAI research is the area of expert systems. Intelligent computer assisted instruction systems and recently intelligent tutoring systems are the names used in the literature to refer to such systems.

As it was mentioned earlier, software used for instruction is referred to as courseware. To develop courseware, one may use a general purpose programming language or a courseware development tool. There are three types of tools for this purpose.
These are authoring languages, authoring systems, and, finally, authoring shells, as introduced in this work.

In Chapter 1 we compared these tools with respect to the degree that they separate code and contents. It is essential to note other differences between an authoring language, an authoring system and an authoring shell. Authoring languages such as TUTOR or Coursewriter are special purpose, high level languages which can be used for creating courseware. These languages are very much like simulation languages such as GPSS and DYNAMO, or artificial intelligence languages such as planner and KRL or LISP which make artificial intelligence programming easier [Kearsley, 82]. Authoring systems are more like expert system building tools. They act as high level interfaces which allow the instructional designer to create courseware without having to learn a programming language. Authoring languages make it easier for programmers to develop courseware, and authoring systems make it possible for non-programmers to create courseware. However, neither of these is applicable to the construction of intelligent tutoring systems. Authoring shells introduced in this work are like expert system shells. They could be used for building intelligent tutoring systems. They separate teaching strategies, code and content. Interest in separation of code and content started after the introduction of MYCIN [VanMelle, 80]. Authoring shells were represented as a pyramid with a changeable tip in Chapter 1. The tip of the pyramid contains the material to be taught, and the bottom of the pyramid is a body of tools. Educators often like to develop their own courseware [Maddux, 92]. We have developed our intelligent assistant with the teacher, as the author of
courseware, in mind. Teachers are meant to be the users of the intelligent assistant for constructing their courseware.

Authoring languages provide a set of primitives for writing instructional programs, and authoring systems make the program as transparent as possible, while authoring shells should automate many of the tasks (e.g. knowledge acquisition) for constructing intelligent tutoring systems.

As stated earlier, different terminology has been used for the same meaning in the field of CAI. In this section we will adopt some of these terms and use them throughout this work. Computer Assisted Instruction (CAI) and Computer Based Instruction (CBI) and Computer Based Learning (CBL) are used in the literature to refer to courseware using the conventional approach. When the computer is used for training then Computer Assisted Training (CAT) or Computer Based Training (CBT) are used respectively. We will use CAI to mean instruction of any type using the computer not employing artificial intelligence and to mean in general, computers used in teaching.

The name intelligent tutoring systems was first used to refer to computer based instructional systems using artificial intelligence techniques in the book edited by Sleeman and Brown [82]. Intelligent Computer Assisted Instruction (ICAI) and Intelligent Tutoring Systems (ITS) will also be treated as synonyms.
2.2 Artificial Intelligence, Expert Systems, and Instruction

Even though computers were originally built as numerical processors, a group of scientists started to explore the ability of computers to manipulate non-numerical symbols. Over the years, individuals concerned with symbolic processing and human problem solving formed a subfield of computer science called artificial intelligence (AI). This goes back to years of 1955-1960, when the computers were becoming more available and general problem solvers (GPS) were developed.

During the late 60's some corporations, thinking that some of the research in AI could prove useful in their corporations, established AI groups to develop practical applications. These efforts were not successful. The reason was that AI programs were too costly to develop, were too slow, and didn't produce sufficiently practical results [Harman and King, 85]. However, AI researchers continued to work in the universities and made various theoretical advances. Significant theoretical advances combined with the development of a new generation of faster and more powerful computers were key factors in the success of AI in the 80's.

However, a more significant breakthrough in AI came with the development of expert systems. Expert systems use large amounts of knowledge in specific domains to solve narrowly defined problems as well as experts in the field can. This can be regarded as a shift in AI from the study of problem solving in the
abstract to the focusing on replicating the behaviour of a specific expert engaged in solving a problem (Harman and King, 85). The first expert system was developed in late 70's at Stanford and it was called DENDRAL (refer to [Sarrafzadeh, 87] for a review of the existing expert systems).

MYCIN [Shortliffe, 76] is one of the most successful expert systems referred to in the literature. However, programs like MYCIN are not usable teaching systems by themselves. They must be augmented with some teaching expertise [Clancy, 79]. GUIDON and its successor GUIDON II are examples of an ICAI system which was developed on the idea of using the knowledge bases and problem solving capabilities of expert systems for teaching. GUIDON is used for transferring the expertise of MYCIN to the student. This means that it is possible to enable an expert system to transfer its expertise to a source other than itself, similar to the way a human teacher would.

Development of many sophisticated expert systems in various fields of application and progress in AI sub-fields such as search methods, natural language understanding, machine learning, automatic text understanding, automated knowledge acquisition and knowledge representation has made the AI methods very attractive for application to the use of computers in education.

2.3 History of CAI Systems
The first generation of teaching systems originated around 1926, before the computers. The course material was a sequence of frames containing explanatory information and a set of related questions. A course was presented frame by frame in sequence without any branching [Gable and Page, 80]. Each student went through the same procedure and answered the same questions.

The second generation was introduced in 1962 and used a so-called scrambled textbook. In these systems a question is presented to the student depending on his/her answer to the previous question. So, branching takes place which depends on the answers given by the student. Many of the systems of this generation do not "know" the subject they are teaching [Gable and Page, 80]. In such systems every statement, question, and branch, has to be specified by the courseware designer.

The third generation, which has been called adaptive systems, differs from the previous generation in that branches are based on the history of answers rather than just the answer to the previous question. This is why a student model has to be built up by the system for each student.

The fourth generation of CAI systems are called generative systems. They use algorithms to generate questions and problems to be asked. This gives the system an artificial understanding of the material. The system has some knowledge which can be used to train the student, and ask questions, and give problems in an unexpected way. In addition to generating problems, generative CAI systems must be able to solve the problems generated [Koffman
and Perry, 76]. Many AI techniques have been used for generation of problems or answers. Early generative systems were used to generate problems in mathematics.

As stated before, by adding some teaching expertise to the existing expert systems, they can be used for teaching. GUIDON was developed on such a basis using MYCIN, which is an expert system used in the domain of infectious diseases. GUIDON is MYCIN plus some tutoring rules, and a model of the learner is constructed by GUIDON [Clancy, 79].

2.3.1 Some Significant Systems in the History of CAI

It was earlier pointed out that there are differences in authoring languages, authoring systems and authoring shells. In fact, each provides a foundation for its successors. The first authoring language in the literature is TIP (Translator for Interactive Programs) developed in early sixties for an IBM 650. This led into the development of another language called the Coursewriter [Kearsley, 82]. Coursewriter was a frame based author language. It used frames, including text related to the subject, questions and the related answers, and branching logic, to organise the course material.

Another language called CATO (Compiler for Automatic Teaching Operations) was developed at the University of Illinois. CATO was used to develop a CAI system called PLATO. PLATO is a system that was used world-wide, for teaching many different subjects in engineering and related fields. Physics, organic
chemistry, computer science, mathematics, theoretical and applied mechanics, architecture, and nuclear, aeronautical, electrical, chemical and civil engineering are among the areas in which PLATO is being used for teaching [Bitzer, 86]. CATO is different from Coursewriter. It does not use the frame concept. Instead, it uses some primitives defining the actions to be taken when a particular key is pressed at the terminal. Some keys are used for defining the instructional logic. TUTOR, a successor of CATO was developed on the basis of frames and still kept many features of CATO. Major vendors active in education have developed authoring languages. Example of these are IDF from Hewlett Packard and DECAL from DEC.

The languages discussed so far are examples of system specific languages. Many system independent author languages have also been developed. PLANIT (Programmed Language for Interactive Teaching) was developed around 1966 and implemented on a variety of machines such as IBM/360, Univac 1108, CDC 6400, PDP-11, VAX and ANGYK-I2. Other machine independent authoring languages are NATAL (National Author Language) developed in Canada for a wide range of computers, and PILOT (Programmed Inquiry, Learning or Teaching) developed for a number of mini and micro-computers.

The first authoring system was introduced in late sixties. It was called VAULT (Versatile Authoring Language for Teachers). VAULT was written in PL/I. It introduced the concept of separating logic from content. This concept was also used in TICCIT (Time-Shared Interactive Computer Controlled Information Television).
Three major types of authoring systems have been designed, namely, macro-based, form-driven and prompting. Mostly, however, they are based on the concept of frames [Merrill, 91].

Macro-based systems give the author choices of high level commands called verbs. Verbs can be used to take care of operations such as presentation of text, processing of student answers, and branching. In this context, the author does not need to worry about many details necessary in author languages.

VAULT is a macro-based authoring system. VAULT has logic and data divisions intended for the user to define instructional strategies and content respectively. Different macros are used for logic and data divisions. An example macro for taking an appropriate action when the student gives a correct answer is shown below [Kearsley, 82].

```
IF CORRECT THEN PERFORM
   DISPLAY MESSAGE A1
   PAUSE 20 second
   IF CORRECT <16 THEN GOTO PREVIOUS-LESSON
END
```

The author uses macro's logic sequence to provide content. Message A1, referenced in the sample fragment above, has to be provided as well. BOOK is a more recent macro-based authoring system.
<table>
<thead>
<tr>
<th>System</th>
<th>Subject Domain</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCHOLAR</td>
<td>Geography</td>
<td>1970</td>
</tr>
<tr>
<td>WHY</td>
<td>Weather</td>
<td>1982</td>
</tr>
<tr>
<td>INTEGRATE</td>
<td>Integration</td>
<td>1982</td>
</tr>
<tr>
<td>SOPHIE</td>
<td>Electronics</td>
<td>1982</td>
</tr>
<tr>
<td>WEST</td>
<td>Arithmetic</td>
<td>1979</td>
</tr>
<tr>
<td>BUGGY</td>
<td>Arithmetic</td>
<td>1978</td>
</tr>
<tr>
<td>WUSOR</td>
<td>Logic</td>
<td>1982</td>
</tr>
<tr>
<td>EXCHECK</td>
<td>Logic</td>
<td>1982</td>
</tr>
<tr>
<td>BIP</td>
<td>BASIC</td>
<td>1976</td>
</tr>
<tr>
<td>SPADE</td>
<td>LOGO</td>
<td>1982</td>
</tr>
<tr>
<td>ALGEBRA</td>
<td>Algebra</td>
<td>1983</td>
</tr>
<tr>
<td>LMS</td>
<td>Algebraic Procdr.</td>
<td>1982</td>
</tr>
<tr>
<td>QUADRATIC</td>
<td>Mathematics</td>
<td>1982</td>
</tr>
<tr>
<td>GUIDON</td>
<td>Medicine</td>
<td>1982</td>
</tr>
<tr>
<td>MENO</td>
<td>PASCAL</td>
<td>1983</td>
</tr>
<tr>
<td>STEAMER</td>
<td>Steamship</td>
<td>1981</td>
</tr>
<tr>
<td>RBT</td>
<td>Boiler Operation</td>
<td>1987</td>
</tr>
<tr>
<td>OBIE-1</td>
<td>Pilot Training</td>
<td>1986</td>
</tr>
<tr>
<td>Caleb</td>
<td>Language Teaching</td>
<td>1987</td>
</tr>
</tbody>
</table>
Form-driven authoring systems provide forms prescribing the information needed to create a course or a lesson. TICCIT is a form-driven authoring system. EAASY (Educators' Automated Authoring System) and AIS (Advanced Instructional System) are other form-driven systems being used in Canada and the USA respectively.

Prompting authoring systems interact with the author for developing courseware. Course-Maker is one such system. The following is a portion of the process of creating courseware using Course-Maker [Kearsley, 82]:

run Coursemaker

>WHAT IS THE NAME OF YOUR COURSE?
Mary

>TYPE YOUR INTRODUCTION, WHEN FINISHED PRESS ENTER TWICE
Read the following Poem:
Mary had a little lamb
Its fleece was white as snow
Everywhere that Mary went
The lamb was sure to go.

>TYPE YOUR QUESTIONS, WHEN FINISHED PRESS ENTER TWICE.
What color was the lamb?

>IS THE QUESTION MULTIPLE CHOICE (1), TRUE-FALSE (2) COMPLETION (3) OR MATCHING; TYPE 1,2,3, or 4.
3

>ENTER A LIST OF POSSIBLE ANSWERS. PRESS ENTER TWICE
white, snowwhite, snowey, snoury

>ENTER A LIST OF POSSIBLE ANSWERS. PRESS ENTER TWICE
blue, green, yellow, red, orange, violet, purple, brown, indigo, puce, beige

Note that every detail has to be provided by the author.

Systems such as SCHOLAR developed in 1970, SOPHIE developed in 1974, and other intelligent systems developed in later years using the concepts of artificial intelligence are summarised in Table 1.1. SOPHIE and SCHOLAR and most of the more recent systems use the concept of student modelling and generate problems using artificial intelligence techniques. Some expert systems used for teaching such as OBIE-1, RBT, Caleb and GUIDON also appear in the literature. There are also other systems not mentioned in this section which are being used or under development.

Some of these systems have been used extensively, but, some have not been put to practical use. For instance, STEAMER, RBT and EXCHECK are examples of those which have successfully been used a lot. On the other hand, SOPHIE, despite its importance, and INTEGRATE, have not seen much use [Fletcher, 87].

2.4 Intelligent Computer Aided Instruction (ICAI)

Although many intelligent tutoring systems have been constructed, they are mostly academic or research lab prototypes [French, 90]. A computer based instruction system based on the AI approach, as discussed in Section 1 of this chapter, employs artificial intelligence techniques to gain an "understanding" of the
material to be taught, and the student. An ICAI system may also use AI in other ways. Examples would be the use of AI techniques in the process of authoring, or for representing teaching expertise, as in GUIDON. A tutoring system that uses AI techniques to achieve its goals will be called an ICAI system throughout this study, although such systems may vary in the extent and purpose of their use of AI techniques.

Among the many ICAI systems implemented, SOPHIE ([Brown and Burton and DeKieer, 82] , [Brown and Burton and Bell, 75]) which was revised three times, used for teaching electronics, SCHOLAR ([Carbonell, 70], [Sleeman and Brown, 82] ) used for teaching geography, GUIDON ([Clancy, 79; 87] ) for teaching medicine, are among the most influential ones in the field of ICAI.

ICAI systems can individualise instruction. This dream of educational scientists can be fulfilled by designing effective student modelling modules for ICAI systems. Typically a student modelling module forms a model of the student by inferring the level of the student's knowledge, and what the student doesn't know, and even in some cases what the student knows incorrectly. Student modelling is discussed in detail in Chapter 6. As an alternative, apprenticeship has been used as a model for interaction with the student, rather than the Socratic method common in intelligent tutoring systems. In such a case, student models are not necessary [Newman, 89].

The knowledge base of ICAI systems contains at least two types of knowledge: domain expertise (or course content), and
tutoring knowledge, which are used for teaching, generating questions and answering questions. ICAI systems are generative in nature, because, the knowledge base is searched every time a question is generated or answered.

2.4.1 Problems of ICAI

Many ICAI systems have been developed or are being developed. These systems try to apply AI techniques to develop more powerful and individualised computer aided instructional systems. The claim of the developers mostly computer or cognitive scientists is not however that these systems have reached the optimal goals. A lot of research is underway and a lot may be expected from future systems.

Some authors, who are mostly instructional technologists, have criticised the movement of applying AI to instruction as being instructionally insufficient and unacceptable by teachers [Rosenberg, 87]. In the following paragraphs we will review some of such comments.

ICAI systems try to employ the current technology rather than trying to fulfil the educational needs. Research reports of ICAI systems are considered to be totally unacceptable and thus the favourable conclusions reached in these reports to be false or unreliable conclusions.

Rosenberg [1987] states that communication between system developers and experts (teachers) and users (students) must be
given more attention than it has been given by most of ICAI researchers. To model tutoring, protocol analysis must be performed. This means that data must be collected concerning actual tutoring situations using tests, surveys, interviews and ethnographic studies. Many of the developers of ICAI systems do not have a protocol analysis and those who have such a thing, take a small number of students into consideration. Besides the implementation considerations, choice of knowledge representation, lack of theories of learning or the use of uni-dimensional approach [Tennyson, 87], single style of learning, poor pedagogy, insufficient data to support positive conclusions are among the shortcomings of ICAI systems discussed. Rosenberg finally claims that 96-97% of ICAI systems have been judged unacceptable and that ICAI is far from what it claims to be able to do. He suggests that developers of ICAI systems should spend more time with educators and students, better testing and analysis should be employed and more cautious predictions about such systems revolutionising the field of education must be made.

ICAI systems use artificial intelligence methods with the aim of improving learning. However, learning and teaching are more complex than considered by the current ICAI technology. This may be due to the fact that ICAI systems have their theoretical basis in computer science rather than educational sciences [Hajovy and Christensen, 87].
2.4.2 CAI Versus ICAI

Intelligent Computer Aided Instruction tries to make instruction using computers more effective. The term intelligent here should not be used as a term which differentiates between CAI and ICAI as stated by Yang [Yang, 87]. However, the basis of the two are different. CAI is more associated with educational researchers whereas ICAI research was initiated by computer scientists. So, the aim of ICAI developers has been to examine the applicability of AI techniques to teaching.

CAI systems do not separate content, information about the student, and teaching strategy into different modules. In contrast, ICAI systems are modular and separate code and content. Such systems usually have common basic components such as, the knowledge base, the student model, the tutoring component and the user interfaces. This makes them flexible and, in addition, generative in nature, because the knowledge base is searched each time to formulate a question or to find the answer to questions.

In CAI, content is structured into tasks and sub-tasks to be taught. In ICAI the content is structured using a knowledge representation method. This method of representation may be one of the existing methods or one which is proposed by the implementers of the system. Only some of the ICAI systems have used task analysis methods. An example of such methods is to identify a set of goals, a set of operators and a set of methods to achieve these goals. This in a different context resembles a method used in artificial intelligence for solving problems known as problem reduction [Nilson, 79].
In CAI student responses are used as a quantitative measure of the student learning progress. In ICAI student modelling has become an area of research and serious work is being performed to find better ways of modelling the student. Various methods for modelling the student are reported. The most important are the overlay model, the differential model and the perturbation model [Sleeman and Brown, 82]. The overlay model represents the knowledge represented in the system which is known by the student. In other words, the student's relevant knowledge is an overlay of the expertise in the knowledge base of the tutor. This gap may then be filled in by the system. A differential model is a comparison of the student and system behaviour under identical circumstances. This way what the student does not understand can be determined. A perturbation model represents the student's misconceptions [Dede, 86]. Student modelling will be discussed in more detail later.

CAI systems have been developed for teaching a variety of subjects, from mathematics and science to natural languages and arts. However, ICAI systems have only been applied to a limited number of areas such as mathematics, programming, electronics and medicine. This may be due to the fact that ICAI developers are mostly interested in the applicability of AI techniques to instruction rather than education itself. Consequently they choose subjects most suitable for the techniques they are trying to apply and understand. For the same reason most ICAI projects do not involve educational people or psychologists in their project development teams. We consider this to be one of the problems of
research in this area. The development process in ICAI unlike CAI is not a system approach (planning, analysis, design, implementation, testing and maintenance [Aktas, 89]) and each system, depending on the project, takes a different approach to development. In addition, most CAI systems have been developed for use on smaller, more available microcomputers whereas ICAI systems usually run on larger, more specific hardware meant for AI applications, and are often implemented using AI languages such as LISP and PROLOG which are more suitable for applying AI techniques. Even though simplified versions of such languages exist for microcomputers today, this has only happened in recent years [Park and Seidel, 87].

Both CAI and ICAI developers claim that their systems individualise instruction, but it is not so for most CAI programs. For a program to provide individualised instruction, it should have several attributes, given by Yang [1987]. A system which provides individualised instruction must treat each student in a different way according to the learner's background, pace of learning, motivation and other factors related to the learner. ICAI systems may be able to provide individualised instruction by constructing a model of the student. Both CAI and ICAI provide for one to one instruction as there is a computer per learner. However, individualised instruction may even be possible with a group of students if the above mentioned factors are taken into consideration.
2.4.3 Related Work

"It is generally accepted that it will be at least a decade before generalised tutoring systems could come into general use" [French, 90]. However, some steps have been taken towards constructing general purpose tools for building intelligent tutoring systems. In this section we will refer to these systems.

ICAI started as an enterprise for using AI techniques to solve the problems of traditional CAI. The issue of knowledge representation has been a central one, and ICAI systems have benefited from developments in expert systems. Expert systems were used in construction of intelligent tutors (e.g. GUIDON). With the introduction of expert system shells, ICAI researchers focused their attention on using them in education. Special expert system shells were developed and used as classroom tools [Valley, 89]. Expert system shells have their own limitations and have not been widely accepted as general purpose tools for constructing ITSs. Along with developments in AI and attempts at employing them in ITSs, there have been isolated and limited attempts at developing instructional shells. We use the term instructional shells to refer to tools which are used for developing ITSs by automating some of the tasks involved. In Chapter 1 we proposed authoring shells which are ideal ICAI system building tools. Although authoring shells with the capabilities suggested in this work are new, there have been attempts at automating certain functions in construction of ITSs. The number of these attempts is small and each is limited and has differing characteristics [Sleeman, 86; 87]. Each of these shells focuses on a certain aspect of ICAI. The first
such system was BIP [Wescourt et. al., 77] developed at Stanford University. BIP controls the task selection process using curriculum scripts and a set of topics to be taught. BYTE [Bonar, 85] a similar system, uses a curriculum network. LMS [Sleeman, 82] concentrates on student modelling and providing remedial material. Lan and Chang [92] have reported a prototype courseware production and presentation system. Their system is an aid in presentation of instructional material. Each of these systems concentrates on a certain aspect of ITS construction and we will not go into their details.

Other work in this line include those of Mizoguchi and Ikeada [91] and Dillenbourg et. al. [92], concentrating on student modelling and proposing general frameworks for building ITSs.

A more relevant work is the recent work of Merrill and his co-workers [Merrill, Li, Jones, 91; 92a; 92b; 93] on transaction shells. This work is of interest to us because it uses a knowledge acquisition tool for automatic acquisition of expert knowledge for construction of ITSs. Recall that a knowledge acquisition tool is a component of the authoring shells proposed earlier in this study.

A transaction shell provides both an authoring and a delivery environment. The authoring environment is meant for courseware development, and the delivery environment is used for the delivery of the knowledge contained in the knowledge base to the student. The knowledge base is filled through the knowledge acquisition component which interviews a subject matter expert. Instructional parameters are used for customising the learning environment.
Although transaction shells are not what we are working towards, they share similarities with authoring shells. The main similarity is the use of a knowledge acquisition tool for acquisition of domain knowledge. We believe that with the current state of art knowledge acquisition it is not possible to have a general purpose knowledge acquisition tool which can be applied to the construction of ITSs in various domains. We do however accept this as a trade off of generality for practicality. Authoring shells proposed in this study, and the tool constructed, make courseware construction easier and rapidly possible. Further, the hypertext environment makes authoring easier and the student modelling component generated by the intelligent assistant eases the burden of constructing ITSs.

2.4.4 ICAI Systems Architecture

2.4.4.1 General

The architecture of CAI systems is not adequate for individualised instruction. Starting in 1970's with the introduction of Scholar by Carbonell [70], a functional architecture for ITSs has evolved. This approach to ICAI architecture assumes certain components, each playing its pre-specified role. This is called the functional approach. The second approach is a bite sized approach [Wenger, 87], which is to have so-called bites, each of which deals with a particular unit of the domain. Such an architecture is meant to be reusable. In the following sections these two approaches will be covered.
2.4.4.2 The Functional Approach

Most of today's ICAI systems consist of four common functional components. These components are: the tutoring module, the knowledge base, the student modelling module, and the user interface, as shown in Figure 4.1. Each of these components and their functions will briefly be explained in the following paragraphs.

2.4.4.2.1 The Tutoring Module

To present the topics and concepts related to each topic to the student, this module has access to the knowledge base containing the domain knowledge. The concept to be presented is selected by this module taking the student model, which is the current state of the student's knowledge, into consideration. This component, which contains an inference engine, must search the knowledge base for problems and must be able to solve them for comparison with the student's answer. A choice of teaching strategy would be desirable in an ICAI system.. Most of today's ICAI systems use a single teaching strategy such as the Socratic method or the mixed initiative strategy. Researchers in ICAI have developed and used techniques for applying various teaching strategies, an example of which is the t-rule approach used in GUIDON [Clancy 1979] The tutoring module should answer a student's questions and solve his/her problems. The tutoring module may use the knowledge base and content in other forms such as text, graphics, to teach the student.
2.4.4.2.2 The Knowledge Base

As important as the knowledge of how to teach is the knowledge of the domain for teaching. As mentioned before, conventional CAI systems did not understand the subject they were teaching but ICAI systems have an artificial understanding of the subject. This is made possible by separating the knowledge of the domain from the inference and control mechanisms of the system. To present a concept, generate questions/problems or to answer/solve them, the knowledge base is searched. This means that the system refers to the knowledge in its possession, while teaching. The system is generative in nature as the knowledge base is searched for generating each question or its answer.

![Diagram of ICAI Systems Components](image-url)
The knowledge base of an ICAI systems contains domain specific knowledge which is formalised in such a way that is understandable (manipulatable) by the system. Normalisation of the knowledge is an important area of AI research. The main approaches to knowledge representation are semantic networks, production rules logic and frames [Sarrafzadeh, 89]. To overcome the limitations and exploit the advantages, we believe that more than one of such methods may be used in combination. The need to represent knowledge is not limited to the knowledge base in an ICAI system. The student modelling component will also need to represent the current state of the student's knowledge, and its relationship to the domain knowledge. Representation schemes for student modelling have been developed. Examples of these are genetic graphs and procedural networks, as explained in chapter 6.

2.4.4.2.3 The Student Modelling Module

The tutoring module uses a model of the learner to adjust itself in order to teach in an individualised manner. Such a model may contain information about what the student knows, what s/he does not know, what s/he knows incorrectly, and information about what the student is currently up to. This is not easy to do. One of the main areas of current research in ICAI is student modelling. Two approaches are common. One is to think of the student's knowledge as an overlay of the knowledge possessed by the system. The second, which is similar to the first, is based on the student's misconceptions. The most significant of these are perturbation modelling and differential modelling.
Student modelling is a specific case of the general topic of user modelling. The current methods are the results of some experimental ICAI system development. There are also attempts at developing stand-alone student modelling systems. Student modelling and user modelling research has made moderate progress towards developing solid principles or practical methods. This is because the field is new and the task is not straightforward. Extracting the correct domain knowledge can be automated, and if it is done manually it can be done through interviews with the expert (teacher). Whereas, automating the generation of possible bugs in a domain is much more difficult. To do this manually one would have to interview the student who doesn't him/herself recognise the misconceptions s/he possesses. This is the reason for development of bug catalogues for some very simple and restricted domains, which is time consuming and usually involves paper and pencil testing.

Using a student model in ICAI can individualise instruction, as instruction can be altered with respect to the student model. Progress in this area of ICAI research can lead us to more individualised instruction and to fulfil the promises of instructional sciences. This will require close co-operation between computer science, cognitive science and instructional technology researchers.
2.4.4.2.4 The User Interface

This is the module that manages the interaction between the student and the system. The domain knowledge must be available to this component in an easily readable or translated form, understandable by the student. In some situations, it is desired that this component understand natural language and to analyse the student's natural language input and to translate it to a form understandable by the system. In other cases, a constrained vocabulary and input style may be more appropriate. Most of today's ICAI systems do not understand natural language. They instead accept a subset of natural language or use other ways of restricting user's input format.

There are a number of interaction styles which may be used in a user interface. These include menus, forms, command language, natural language, and direct manipulation. Little training is required for menu selection and the number of keystrokes is minimal, but possible disadvantage is the slowness of wading through menus, and the amount of screen space they occupy. Forms with blank spaces are easy enough to tab through, but guidance or training may be required on permissible values for fields. Command languages are expressive in the hands of experienced users, but novice users require training and have high error rates. Natural language presents interpretation problems, which may have to be resolved by further dialogue, and is often more voluminous for the user to enter.

Sometimes objects of interest can be represented visually, and the user can directly manipulate these objects. The Macintosh OS, most video games, and aircraft control systems use this style.
Some CAI systems also do [Burger, Desor 91], while LOGO [Papert 87] is another educational example. [Davenport 91] describes Window /Mouse interfaces in some CAI systems.

Depending on particular needs, one or more of these styles may be combined. Commands may take the user to pop-up menus or forms. Natural language may be combined with menus, as in NLMENU [Tennant et al., 83]. User interface management systems are useful tools in this context.

Many of the ICAI systems are game coaches where the input is a set of moves not really requiring natural language processing. Some of them deal with subjects which are mathematically oriented or have their own special language. A very few systems, like SOPHIE, provide for natural language input. The topic of natural language understanding is an area of research influencing ICAI research, and progress in such research will surely improve the performance of ICAI systems.

2.4.4.2.5 The Bite Sized Approach

This approach to ICAI systems architecture is based on pedagogical issues. Such an architecture consists of a collection of bites containing domain knowledge, relationships among bites (conceptual, curricular, etc.), the student's knowledge state related to this bite's piece of knowledge, problem generation mechanisms and other related information. Functions such as student modelling, inference mechanisms and others are distributed in the architecture. It is claimed that using such an architecture allows the use of parts of the architecture for other domains, depending on its closeness to the domain of the existing
system [Wenger, 87]. This approach to ICAI systems is not commonly used. The functional approach is more common. The idea of authoring shells, as expected, addresses the issue of reusability. Bite sized architecture is supported by the intelligent assistant constructed in this work. The bites in our case are the concepts, and all the information necessary for knowledge representation and student modelling are stored for each concept separately, as explained in the next chapter.

3. Representing Domain structure Using the Concept Relationship Model

3.1 Representing the Domain

Knowledge may be viewed as a body of information and skills acquired by people. It can be transferred, stored, retrieved and processed. The first ingredient of intelligence is knowledge. Thus, an intelligent system must possess the necessary domain knowledge. Studies have been conducted on how humans represent knowledge, mostly in cognitive science. We will examine these studies in this chapter and indicate how they relate to our study. Minsky in a keynote speech at the IJCAI-91 conference in Sydney, stated that he believed humans use different ways of representing knowledge for different types of knowledge. This belief is also shared by this research. Computer representation of knowledge is an important topic of research in artificial intelligence. Knowledge representation is an important sub-field of artificial intelligence. Sowa [84] defines artificial intelligence as "the study of knowledge representations and their use in language, learning and problem solving ". Linguistics and cognitive science researchers
are also among those involved in research in knowledge representation. Knowledge representation languages are used for representing the knowledge of a specific domain. The most commonly used formalisms are logic, production rules, frames and semantic networks. A formalism may be graphical (e.g., semantic nets) or symbolic. Knowledge represented using one of these formalisms can be converted into other formalisms. For instance, one may convert a piece of knowledge represented using a semantic network into first order logic clauses. It is, however, sometimes more natural or more convenient to represent a certain chunk of knowledge in a particular representation formalism. This is why we believe that a combination of these formalisms used in a system is advantageous over using a single formalism. Figure 3.1 depicts this fact in the context of this work. Semantic networks were introduced and implemented in a computer program in 1966 by Quillian, a student of Newell and Simon.

3.1.1 Knowledge Representation for ITSs

Intelligent tutoring systems, being artificially intelligent systems, have knowledge representation requirements. Several different types of knowledge are required in an intelligent tutoring system. Knowledge about the structure of the domain is needed. This is what we call structural knowledge. Less attention has been devoted to this type of knowledge in the past. Knowledge about the student and perhaps the teacher is another type of knowledge which is necessary. For student modelling, we have adopted the overlay modelling approach where the knowledge kept is the knowledge of what the student's state of knowledge is.
Otherwise, knowledge of the student's misconception would have been necessary. Based on a student's knowledge, plans may be constructed. Planning knowledge is yet another type of knowledge necessary in ITS. Teaching strategies must somehow be represented in the machine. Domain knowledge, which is the subject knowledge being taught is also required. In other words, each of the components of the architecture of ITS explained in Chapter 2 has its own knowledge representation requirements. The student modelling component requires knowledge about the student's understanding of subject matter. This may sometimes also include knowledge about what the student knows incorrectly. The knowledge base containing domain knowledge has general knowledge representation needs. Different domains may have differing knowledge representation requirements. Traditional CAI systems have knowledge of the subject they teach. They are, however, unable to reason with this knowledge. Some ITSs interact with the student using natural language (or a subset of it eg [Bonar, Cunningham, 1988]). This adds another type of knowledge to the list of an ITS for use in the user interface. The tutoring component would need to both represent teaching strategies and plans for presenting content to students. Planning is representation and manipulation of goals [McCalla, 87]. In addition to the techniques used in construction of an ITS, research in knowledge representation for ITS has had some results. Examples of this are the genetic graphs of Goldstein [82], and Burton's [82] procedural networks discussed in Chapter 5, and other formalisms such as feature networks [Webb, 88]. General knowledge representation formalisms have also been used in ITS. For instance, semantic networks were used in the first intelligent
tutoring system SCHOLAR [Carbonell, 70]. They were used with the assumption that human symbolic memory is similar to semantic networks production rules, as used in GUIDON [Clancy, 79; 86] to represent both domain knowledge and teaching knowledge (t-rules). In the PROUST automatic debugging tutor [Johnson, Soloway, 85], frames were used to represent plans and goal.

3.1.2 The need for Multiple Formalisms for Representing Domain knowledge

It is clear from the examination of ITS and the formalisms they use for representing knowledge that a single formalism would not fulfil the knowledge representation requirements of such systems. In the general tool we have suggested in this work there must be provision for multiple knowledge representation formalisms. At the bottom of the pyramid in Figure 1.2, there are tools for manipulating knowledge represented in different formalisms. In the descriptor of each concept in the CR network, the knowledge representation requirements of that concept are stored. This idea is depicted in Figure 3.1. Storing this information in the CR network will allow the use of the designated formalisms and will make the necessary links between that concept and the corresponding knowledge manipulation tool.

3.2 Relevant Cognitive Psychology Research

To elaborate on the importance of structure, on which I have placed a particular emphasis, it seems instructive to look into
some of the relevant psychological views and theories in cognitive psychology. Of the most importance in cognitive psychology is the general tradition of constructivism. Some of the most important thinkers of this tradition are Piaget, Kelly, Bruner and Ausobel.

Generally, constructivism is a view that puts the main emphasis on the mind rather than environmental elements. That is to say, the mind’s ability to impose constructs or structures on environmental stimuli is what constitutes the core feature of constructivism. Constructivism views cognition as proactive in which human knowing occurs in an active and constructive (form-giving) way [Mahoney, 88]. This view rejects passive conceptions.

---

**Figure 3.1 Multiple Types of Representation and Multiple Formalisms Needed in ITS**
about the nature of mind, according to which the main role of the mind is to receive information from environmental stimuli and store them as copies.

Piaget as a developmental psychologist has shown the constructive role of mind in the genesis and development of intelligence. This constructive role of mind is best exemplified in his notion that intelligence organises its world by constant self-organising.

A lot of work has been done by cognitive and educational psychologists on mental development, concept acquisition, and learning. Such work is mostly based on the mental development stages theory of the famous psychologist Jean Piaget [Piaget 1962] who suggests that humans go through certain stages of mental development. The final stage in Piaget's mental development model is the formal operations stage. At this stage, one has the ability to deal with abstract ideas. The important point in this stage is what Piaget called hypothetical-deductive thinking. This is a context-free (formal) type of thinking and representation. At the earlier stages, the child's thinking is concrete and cannot separate form and content. That is to say a child in this stage cannot think in an abstractive manner. At the formal operations stage however, the individual is capable of this. The model we have developed here is thus only applicable to individuals in their final stage of development.

The stages theory of Piaget is frequently referenced by ITS and CAI researchers. Piaget's model of how knowledge is
represented is of more significance to this research than the stages of development, although the mental development stages are taken into consideration. The staged view of mental development is known as developmental psychology. In cognitive psychology, the representation of knowledge is of prime importance, whether in the concrete or abstract stages of development.

Cognitive psychology is, "the scientific study of human mental processes and memory structures in order to understand behaviour [Mayer, 1981 ]. Cognitive psychology focuses on how information is acquired, how it is stored, and how it is retrieved and processed when necessary. Cognitive psychology views humans as active processors of information [Sewell, 90 ]. In this view, learning is coding according to meaning and is called semantic encoding. Meaning related to sensory activities (e.g. Hearing an alarm) is held in what is commonly known as a short term memory store. This information is not to be kept for a long period of time. Information that is to be held longer and not to be forgotten is put into long term memory. This is done through semantic coding, i.e. according to meaning. A semantic network is a model of how information may be stored in long term memory.

Bruner and other cognitive psychology researchers have placed much importance on how information is represented in human memory. This view implies that the structure of a subject is of prime importance in teaching and learning of it. We have based our work on this fact and used it for developing new
techniques in automatic hypertext generation and courseware preparation.

Following Piaget, mental development research was continued by researchers such as Gagne and Bruner. Gagne suggests a hierarchy of eight types of learning, each more complex than the previous one. Bruner assumes three modes for human representation of the environment, and events in it [Graves, 1983; Naish, 1982]. Bruner [1960] clearly points out the importance of structure in learning. He views relationships as being of prime importance in grasping the structure of a subject. He states that "grasping the structure of a subject is understanding it in a way that permits many things to be related to it meaningfully. To learn structure in short is to learn how things are related".

Ausubel's cognitive learning theory and his differentiation between rote learning and meaningful learning are of interest to this work. Meaningful learning in this context takes place when new knowledge is integrated into an existing network of concepts and principles in the cognitive structure. According to Ausubel, the individual's motivation in learning is mainly controlled by the meaning s/he might find in what is to be learned.

This in fact supports the validity of our model as we have applied it to courseware preparation and for that purpose, student modelling. An individual's cognitive structure, to an extent, mirrors the instructional content acquired during instruction. Several techniques for representing cognitive structure have been used. These techniques were used, both to assess the students'
learning before and after instruction, and to assess the degree to which cognitive structure mirrors subject matter [Stewart, 1979]. CR networks can in fact be thought of as another way of representing content as well as cognitive structure. The difference between our method and previous methods is that it is more abstract and it is constructed independent of content.

Concept mapping seems to be the main model used in cognitive learning studies. In the following sections, we will briefly review previous work in concept mapping and how it relates to this study.

3.3 Concept Mapping

Concept mapping has been used in classrooms at different levels from primary school to tertiary level. Concept mapping has been suggested as a useful tool for enhancing learning when textual material is used [Dansereau 78; Dansereau et. al. 79a; 79b; Novak, Gowin, Johnson 83; Anderson 79]. Feifer [92] has constructed an ITS for teaching concept map construction from text.

Although concept maps are much more specific than our CR networks we will discuss them briefly here. They are actually closer to semantic networks than to CR networks. Concept maps are another way of representing domain knowledge and they are yet another type of semantic network, or another application of semantic networks. They do not really represent the structure of the domain. Concept maps are usually used to represent the
knowledge contained in a passage from an article, a page of text, or a matter discussed in classroom. Ghaye [84] has studied concept mapping and has analysed a number of pupils concept maps. He has classified concepts and relationships that are possible. Ghaye uses concept maps as one kind of evidence of pupil learning in actual classroom situations.

Concept maps are used for representing a pupil's cognitive structure. This cognitive structure, even in relatively small domains, can be complex and it is unique [Van Den Daele 69]. Several researchers have studied the relationship between the content structure (represented subject matter) and cognitive structure [e.g. Shavelson 71; Preece 76a, 76b, Rudintsky 76]. Any method used for representing and assessing cognitive structure should allow for comparisons to be done before and after instruction, as suggested by Stewart [79]. This is what student modelling is actually all about and this is why we view concept mapping as being useful. Different methods of constructing concept maps have been used [e.g. Shaefer, 79; Wandersee, 83; Ghaye, 84; Robinson, 86; Novak, 77]. Concept maps have been used in social science but have seen less use in mathematics [Malone, Dekkars, 84]. Novak [1981] defines concept mapping as a process of identification and organisation of concepts into a hierarchical arrangement. Most concept maps have in common the concepts and the relationships among these concepts. Figure 3.2 shows an example of a concept map. Concepts are usually represented as boxes or ovals and relationships are represented as lines. These lines are sometimes unlabelled [Moreira, 79; Cliburn, 86] leaving
out an important part of concept mapping which is the relationships binding the concepts being mapped.

Robinson [86] has found concept mapping to be of great value in structuring of geographical knowledge. Many concept maps were drawn by students and analysed in her study of concept mapping as a tool for teaching of geography. The structure found in the concept maps was classified as follows:

- Random: isolated concepts
- Disjoint/paired: pairs of isolated concepts
- Linear: linearly joint
- Cyclic: a cycle exists
- Linear cyclic: combination of linear and cyclic
- Radial: linear chains connected to a central concept
- Combination: a combination of all the above
- Hierarchical: a hierarchy of concepts
- Complex: many concepts are linked
  - Simple: more concepts than links
  - Advanced: more links than concepts
- Hierarchical/complex: hierarchical structure retained while many concepts are linked

Most concept maps are hierarchical, in spite of the categorisation of Robinson. This categorisation has been summarised in Figure 3.2 from Robinson.
Analysing a large number of concept maps drawn by students, Robinson [86] classifies the links between concepts into four categories. These four categories of links are:

1. Taxonomic
   - logical
   - descriptive
   - belongs
   - comparative

2. Functional
   - functional
   - functional action
   - choice

3. Action
   - action-doing
   - action-causal

4. Procedural
1. RANDOM:

2. DISJOINTED/PAIRED

3. LINEAR

4. CYCLICAL

5. LINEAR-CYCLICAL

Combination of 3 and 4

Figure 3.2 Classification of Concept Maps
Figure 3.2 Classification of Concept Maps (Continued)
3.3.1 A Procedure for Constructing Concept Maps

Concept maps are usually drawn by the student. Procedures [Malone and Dekkers, 84; Ault, 85] and methods for teaching concept mapping have been devised so that students can easily draw concept maps. Concept maps have been referred to as windows to the minds of students [Malone, Dekkers, 84].

Ault [85] proposes a simple set of techniques for construction of concept maps and suggests that they must be attempted to be understood. He has broken the task into 5 steps.

1. select: Material for mapping is selected in this step. This material may be a text passage, lecture notes or the like. The material is to be read and a central concept and then other concepts found by selecting words, phrases, objects and events.

2. Rank: Rank the concepts identified from abstract to specific. This will put concepts in a hierarchical order.
3. Cluster: Group concept at the same level of abstraction and interrelated concepts together.

4. Arrange: Use a two dimensional array to arrange concepts with respect to their neighbours.

5. Link: In this step, concepts are linked and links are labelled. This is done with a pair of concepts at a time.

Stewart [80 ] has proposed a procedure for converting sentences into a concept map. We have simplified it as follow:

Repeat for each sentence

1. Analyse the sentence to identify the concepts used in it

2. Put these concepts on a piece of paper and enclose them in boxes.

3. Connect the boxes where necessary and label the lines with the corresponding relationships.

end repeat

The arrangement of boxes and lines may be changed as the concept map is constructed. Once completed, a concept map can be used to analyse the student's understanding of the material. Both prior knowledge of the students - by comparing a group's concept maps - (i.e. Overlay model) misunderstandings of students (i.e. buggy model) can be identified.
Concept maps have been used in a variety of forms. They have been used mainly by educators as a tool for assessing the extent to which students have learned what they have been taught. They are thus a good tool for modelling the student's understanding of subject matter. They are very similar to semantic networks. Concepts are usually very specific. Concept maps are a domain knowledge representation formalism suitable for student modelling. Carbonell [79], a pioneer in ICAI, used a semantic network which was marked as the student learned. Concept maps are usually used for converting small pieces of content, such as a passage from an article or a page from a textbook or lecture notes, into a structure which is matched by the structure that would be formed in the long term memory when one learns the same material.

3.4 Semantic Networks

"Semantics is the science of meaning." [Carbonell, 70]. In different contexts, semantics is interpreted to refer to different things. In linguistics, for instance, semantics deals with the deep structure of sentences. Syntax on the other hand deals with how words are sequenced in a sentence. Semantics resemble the way human long term memory stores information [Carbonell, 70].

The idea of modelling human memory based on association can be traced back to Aristotle [Brachman, 1985]. It was, however, Quillian [1966], a student of the famous Newel and Simon, who introduced a network model of human memory called semantic
networks and implemented it on the computer. Quillian produced a simulation program for processing English language sentences.

A semantic network is a network of nodes connected by arcs. The nodes are objects and the links are the relationships between the objects. The most common relationships are the is-a and has links. The is-a relationship depicts inheritance and the has relationship depicts the object-attribute relation.

Quillian's semantic networks were a reflection of the model of memory he had constructed [Bench-Capon, 90]. They consisted of words and how they were related to other words representing the meaning structure of the language. Many network based formalisms have appeared in the literature and have been labelled semantic networks [Carlson, Ram, 90]. Scholar [Carbonell, 70] an early ITS, used a semantic network to represent its knowledge. This knowledge is specific knowledge in the domain of geography. The semantic network was marked as the student learns. This was done for the purpose of student model construction in Scholar.

3.5 The Concept Relationship Model

In this study we introduce and utilise a graphical knowledge representation methodology called the concept relationship model (hereafter CR model) which has been used as a basis for construction of an intelligent assistant for courseware preparation. As by-products, student models are generated and hypertexts are automatically generated.
A CR model imposes a structure on the knowledge domain. Knowledge about the material related to each concept is kept in a CR model for every concept identified in the domain. The position of each concept within this structure is depicted by its location in the network and the hierarchy and difficulty level information embodied in the model.

The intelligent assistant is based on an abstraction of the domain for which we are constructing a course. This model helps the intelligent assistant retrieve resources available to it from a mass storage device. The model we have developed is the concept relationship model. A concept relationship model is a model of the world constructed for the purpose of retrieval and organisation of subject matter. This model is also useful for assessing a learner's understanding of the structure of the domain. This is used as one source of assessing the learner's knowledge which leads into a model of the learner.

Concept relationship networks are used for modelling the structure of a domain. It is through this structure that we construct a network of domain knowledge in various forms. A concept relationship network (CR network) is a network of concepts and relationships between them. A node in this network may be a concept node, represented as an oval, or a relationship node, represented as a rectangle. Each node points to a record containing information specific to that node. We refer to this information as the node descriptor. The descriptors are constructed by a knowledge engineer through interviews with a subject matter expert.
The best experts in educational settings are the teachers, and a knowledgeable and experienced teacher must be selected. It is possible that another teacher using the tool in a classroom setting may not agree with the structure imposed by the existing CR network. It is also possible that a student may see the structure of a domain in yet another way. The clash between teacher and expert poses a serious problem, that between a expert and student can be resolved at the time of testing, as explained in Chapter 5.

3.5.1 What is a Concept?

A concept is an abstract entity which represents a potential cluster of information. This information can be text, graphics or other media such as video or sound, all linked together. Media is not an issue of concern here, although these resources must be indexed. Concepts are not atomic but may be specific or general. A concept is defined in terms of one or several sets of key words. Concepts are graphically represented as ovals in a CR network. Two concepts may or may not be related in a network. If related, a relationship node will hold information about this relationship. A specific concept has a variety of interlinked resources associated with it. It will not however be expanded further into another CR network. A general concept on the other hand, may have interlinked resources associated with it and will expand into another CR network. An example of a specific concept would be the concept of climate in Figure 3.3. An example of a general concept would be the concept of geography as shown in Figure 3.4 (page 76). The concept of geography can have general introductory information in the form of text, graphics, etc. What is more important is that it expands
into a CR network, nodes of which may further expand into CR network resulting into a multi-layered structure. Although CR networks are not hierarchical structures, each concept within a CR network is given a hierarchy level value. This, together with a difficulty level also associated with each concept, is information that is to be used along with other information, by the target authoring tool to organise the lessons to be constructed. Concepts

Figure 3.3 A Portion of a CR Network

are identified through interviews with a domain expert. They are not dependant on the material used for instruction.
3.5.2 Components of a CR Network

Figure 1.5 shows the general form of a CR network. Figure 3.3 depicts a more specific view of CR networks. A CR network consists of 4 main components:

- Concept nodes
- Relationship nodes
- Descriptors
- Resource links

A concept node is represented as an oval. Relationship nodes are represented as rectangles as shown in the figures. Each concept and relationship has a descriptor. Each concept has a resource link which links it to its resources, and the resources are linked back to their corresponding concept. Resource links are generated at run time.

Concept descriptors are the most important components of a CR network. They play an important role in automatic retrieval, linking, and organisation of course material. More time is spent constructing and revising concept descriptors than in any other part of CR network construction. A concept descriptor has the following essential components:

- Name
- Hierarchy level
- Difficulty level
Sets of keywords
Weights of keywords within each set
Key knowledge
Synonyms
Related concepts.

The name of the concept is recorded for obvious reasons. The expert, at some stage in the development of the CR model suggests a hierarchy and difficulty level for each concept. This information is used for organisation of material and student modelling, along with other information and procedures. Each concept is assigned one or more sets of keywords. These keywords must be selected carefully, as they define the concepts. Weights are assigned to each of the keywords within each set. The weights assigned to keywords in a set must add up to 100. The weight of a concept is an indication of its importance. Synonyms of a concept are also recorded in the descriptors. Another important descriptor item is key knowledge. Key knowledge is acquired from the expert, formalised and recorded into the descriptor of each of the concepts. Key knowledge is a minimal chunk of knowledge reflecting the main ideas of a concept. It is this chunk of knowledge that is used as a source of evidence for student understanding of a concept.

A CR model will at all times contain the abovementioned information. Once retrieval takes place, lines of text from the retrieved documents will be inserted into the descriptors of concepts. This information is used for a customised student modelling component, along with other information. Two other
fields of information also exist in a concept descriptor. These are the knowledge representation formalisms and media types. The knowledge representation formalisms are to support the use of different formalisms to represent the knowledge related to a concept. This was explained earlier in this chapter. The intelligent assistant we have constructed around the CR models can support different media provided that they are indexed. The media types field in the concept descriptors supports multimedia expansion of the system. Once other media are included, the automatic linking capability of IACP can construct a hypermedia using the different media. Note that all this is made possible through the CR model we have developed in this study. Research in cognitive psychology supports our ideas, as discussed earlier.

Each relationship node has a descriptor associated with it. The relationship descriptors are less complex than the concept descriptors. Most of the information in these descriptors is obtained by intersecting certain items of the concept descriptors. A relationship descriptor has the following information items:

Name
Keywords in Common
Synonyms
Related concepts
Strength

The name of the relationship and its synonyms are recorded in the descriptor. Related concepts can be recorded from the graphical representation. Keywords in common is an intersection of the
keywords of the related concepts. The most important piece of information recorded in a relationship descriptor is the strength of the relationship. This is a value between 0 to 1, corresponding to five degrees of strength of very weak, weak, related, strong and very strong. Initially, this degree of strength is assigned by the expert. The strength can be modified through parameter learning as explained in Chapter 3. This modification of the strength values cannot be done until the system has been used several times with the same CR model.
Figure 3.4 A Multi-layered CR Network
Figure 3.5 Summary of the Node of Development from a Higher Level Network

3.5.3 Multi-Layered CR Networks

In principle, multi-layered networks are possible in the CR model. However, we have not constructed a multi-layered network in this study. The network constructed in this study (see Appendix A) can be thought of as an expansion of one single node in a higher layer which may yet be a single node in an expansion of a node of a higher level network. We have summarised this idea in Figure 3.5.

As mentioned earlier, a CR model of a specific domain used for retrieving and linking appropriate material to be used for
teaching or for use in construction of a course using an authoring tool is explained in Chapter 6. CR networks of different domains can be connected at a higher level. This would, however, result in a complex network, construction of which would require time and group effort.

A possible multi-layered network leading to the network of development in geography is depicted in Figure 3.4.

3.6 Knowledge Acquisition

The most significant problem in the construction of knowledge based systems is knowledge acquisition. Put in simple terms, knowledge acquisition is the process of transferring knowledge from the source(s) of knowledge into a machine manipulatable representation. The sources of knowledge may be books, manuals, videos or other similar sources. For much of the information, the source of knowledge is one or more human experts. The knowledge of the expert is extracted by a knowledge engineer and translated into a form manipulatable by the machine. This involves interviewing the expert in one form or another. Direct interviewing seems to be used more often than not. Protocol analysis is also common in knowledge acquisition. Protocol analysis, although more formal, is still a form of interviewing. Protocol analysis is the observation of an expert while s/he solves problems. The expert is usually asked to think aloud. The protocol constructed by the knowledge engineer, of the expert's problem solving activity, is then analysed. In direct interviewing, the expert answers questions posed by the knowledge engineer. There
are no formal methods for conducting the interviews. However, some informal methods are used in such interviews. A technique for knowledge acquisition which is well worth mentioning is the repertory grid analysis method. This method is based on Kelly's [1955 ] personal construct psychology. Repertory grid analysis was used in AQUINAS [ Boose, 88 ], an automatic knowledge acquisition tool. Using grid analysis, a grid of elements and constructs that describe an expert's perception of a problem is set up. Using patterns and associations in the grid, the knowledge engineer causes the expert to focus on key issues [Hart, 86 ].

Because both the knowledge engineer and the expert are human, the task of knowledge acquisition is often the bottleneck of expert systems construction, and the qualities of both parties affect the outcome. The experts' ability to express their knowledge, communications skills, enthusiasm and belief in the project, and willingness to cooperate, and interviewer familiarity with terminology used by the expert are among the factors affecting the outcome. Some of the problems we faced in this context will be discussed later in this chapter. The knowledge engineer does not usually have the choice in the selection of the expert, as experts are a scarce resource and hard to find. This makes the job of the knowledge engineer a critical and tedious one. The knowledge engineer's experience and qualities play an important role in knowledge acquisition. It is expected that the interviewer will be an experienced knowledge engineer with some familiarity with the domain. However, this may not always be the case. Familiarity with knowledge engineering in similar domains is important, and affects the time spent and the validity of the
knowledge extracted. To make knowledge acquisition more effective and reduce the amount of time spent, tools for automating all or parts of the process of knowledge acquisition [Boose, Gaines, 90], or for refining the acquired knowledge, have been constructed. TIERESIAS [Davis, Lenat, 82] is an example of such tools. Although no formal methodologies exist for performing the interviews with the expert, general guidelines have been proposed [Hart 86] which we will summarise below:

Be specific not general about specific things and let the expert talk about cases of interest to him/her.

Do not force the expert to produce specific representations. Let the experts represent their expertise in ways they are comfortable with. This means using graphs, tables, diagrams and the like and not insisting on the use of tools like flowcharts, etc.

Do not interrupt. Be patient and try not to interrupt the expert, even in the face of repetition.

Record the information. If the expert is not annoyed, the interviews should be taped or video recorded and played back and used later.

Listen to the way the expert uses knowledge. This will reveal the way the knowledge is manipulated by the expert, as well as information such as the order in which problems
are being approached, and the importance given to certain pieces of information.

As stated earlier, the qualities of the knowledge engineer affect the outcome of the knowledge acquisition. Some of the important qualities may be summarised as follows [Feigenbaum, McCorduck, 1984; Welbank, 1983; Sarrafzadeh, 94]:

Clear and logical thinking ability: The expert may at some points be confused, inconsistent, or provide invalid information. The knowledge engineer must be able to detect such problems. This may occur more at the early stages in the knowledge acquisition process when the expert and the knowledge engineer are more likely to misunderstand each other's terminology. This problem may be more severe when the knowledge engineer is less familiar with the domain, or the expert does not understand the expert's, or both.

Enthusiasm and belief in the project: It may take some time to reach any results and there may be many gaps and inconsistencies. The expert may lose enthusiasm at some stage. The expert must persist and make sure that the ultimate results are consistent.

Patience: The knowledge engineer must be able to exercise a lot of patience.

Innovativeness and versatility: The knowledge engineer must use his/her own judgement in selecting what seems to be
appropriate. Because of the lack of formal methodologies, much of the work depends on the knowledge engineer who may have to invent methods or representations depending on the domain and the expert. This requires versatility and innovation on the part of the expert.

Good communication skills: Because a lot of time is spent discussing important domain specific matters, effective communication skills are of prime importance. A knowledge engineer must be able to use language skills, use graphical representation techniques and even interpret body language.

Familiarity with terminology: Although in most cases the knowledge engineer is not familiar with the domain, an understanding of the terminology is a big plus.

Programming skills: The knowledge engineer must be familiar with various knowledge representation formalisms, and should also have an understanding of ways of implementing the system.

All these qualities may not be present in all knowledge engineers. These factors, however, affect the success of the project. Several attempts have been made at automating the process of knowledge acquisition. Many tools for aiding in the process have been constructed. Some of these tools are specific to certain domains. The expert would still be questioned by the program. Thus, some of the problems related to the expert will be present. Some of the
knowledge acquisition tools may be listed as follows [Boose, Gaines, 90]:

ACQUINAS, 1987
ETS, 1984
FIS, 1987
Kitten, 1987
Knack, 1987
KSSO, 1987
MDIS, 1983
MOLE, 1987
MORE, 1985
MUM, 1987
OPAL, 1987
SALT, 1988
TIERESIAS, 1982
TKAW/TDE, 1987

Other more recent automated tools include PM [Reynolds et al., 90], Taql, used in Digital's SOAR, [Yast93], KNAMS [Raghuthan 93], each of which has been used for knowledge acquisition in a particular domain. Work on metalevel tools for generating knowledge has recently attracted researchers, see for example DASH [ Eriksson et al.,94].

Some of these tools address a specific domain and some are domain independent. Some support a certain stage in the life cycle of knowledge acquisition and some support the entire life cycle. Automated knowledge acquisition tools are one of the building blocks of the authoring shells proposed in this study. However, we
believe that tools with the generality and effectiveness needed in ITS building requires further progress in automated knowledge acquisition research.

Existing tools have mostly been developed for a specific domain. Most are research oriented, some are used internally in a company, and a few like Autolntelligence from IntelligenceWare are available for purchase. Most are based on a specific technique such as the repertory grid, or automatic interviewing. A few, such as Kriton [Diederich et al., 87], try to provide integrated environments for knowledge acquisition. The main problems which still need addressing are as follows:

1. More testing is needed to prove the research tools in contexts different from the one in which they were developed.
2. The cost of developing such tools is already high, and more extensive proving will make them even more expensive.
3. The knowledge acquired by a tool may be required in a different representation from that in which the tool collects it. This will require transformation rules to be built into the tool, similar to the way that wordprocessors can save files in different formats.
4. General tools do not exist, and there are some areas for which knowledge acquisition tools do not exist at all [Kitto and Boose 89].

3.6.1 Knowledge Acquisition Life Cycle

Figure 3.6 shows the stages in the development of expert systems. The steps that the knowledge engineer(s) has to take to build the system, namely: identification, conceptualisation,
formalisation, implementation, and testing are not well defined and "they are simply a rough characterisation of the complex activities that take place during knowledge acquisition" [Hays-Roth et al., 83]. It is not possible to define a standard set of steps to be followed which will result in optimised expert systems.

In the identification stage the most important aspects of the problem are identified and understood. The terms and key concepts have to be classified at this stage. The end product of this stage is an informal description of the problem.

During the conceptualisation stage the concepts identified during the previous stage must be defined explicitly. The sub-problems and information flow characteristics are defined next. Graphical tools may be used in this stage to diagram the concepts and their relations.

In the formalisation stage, depending on the framework of the tool, the available concepts, relations, sub-problems, and the information flow characteristics, are put in a formal form. The most commonly used knowledge representation formalisms are first order logic, semantic networks, and frames. Depending on the tool or language selected for implementation, one of these or
perhaps another formalism must be chosen for representing the results of the previous stage. The end product of this stage may be the description of how the problem can be represented in the chosen tool environment.

In the implementation stage, the results of the formalisation stage are represented using the language provided by the chosen tool. At this stage a prototype system is built, and is ready to be run and tested by the end of this stage.
The prototype system is evaluated in the testing stage. Example problems are solved by the system in order to find the holes in the knowledge base as well as any problems with the inference structure. Changes and revisions may be needed in this stage, and the knowledge engineer may decide to go back to the previous stages. This implies that the life cycle is iterative and non-linear.

### 3.7 Structure Elicitation

Knowledge acquisition for the construction of an ITS is different from general knowledge acquisition. The difference is that in addition to domain knowledge, the need for a meaningful curriculum of instruction adds to the difficulty.

We view knowledge acquisition in general, and especially as it applies to ITS, as involving two processes. One is structure elicitation and the next is the actual acquisition of knowledge from the sources of knowledge with respect to the structure. The sources of knowledge may be varied, including books, tables, figures, and one or several domain experts. This two staged view of knowledge acquisition is appealing to ITS in general and this study in particular. We briefly mentioned some of the efforts in automation of knowledge acquisition earlier. Automating structure elicitation does not seem to be difficult to achieve, using the CR model developed in this study as a basis. Automations of this type are, however, beyond the scope of this study, but we have developed a method for manual structure elicitation by
interviewing a domain expert, and applied this method to the domain of geography.

Both structure elicitation and knowledge acquisition have knowledge representation requirements. We reviewed knowledge representation and its importance to ITS research. We also introduced a model which we have called the concept relationship model to fulfil both the knowledge representation needs of the structure elicitation process and other purposes which have been discussed elsewhere in this work. Using the structure elicitation methodology introduced here, it is possible to construct a concept relationship network in a specific domain, which represents the structure of the domain. We will describe this methodology in the following section.

3.7.1 A Methodology for Domain Structure Elicitation Using the CR Model

To represent the structure of the domain of interest, we construct a CR network for it, through interviews with the expert. Figure 3.7 shows the non-linear life cycle of structure elicitation. It involves the identification of concepts, establishing of the existing relationships and construction of descriptors for them. The construction of descriptors is repeated for each concept identified [Sarrafzadeh, 93].

The first step in the construction of a CR network is the identification of concepts. In this step the expert is asked to name a key concept of the domain. This concept is written down and the
expert is asked questions leading to identification of other concepts until no other concepts can be identified. This may take several sessions. In each session the results of the previous session are reviewed. After the last session in identification of concepts, the expert is presented with a list of identified concepts and asked to assign a level of difficulty on a scale of 1 to 10 to each concept and assign synonyms where possible to each concept. The expert may add new concepts or delete combine or rename concepts while assigning difficulty levels and synonyms. Questions must be worded carefully and the results of previous sessions must be studied by the knowledge engineer before the subsequent sessions. Where possible, problems should be discussed with the expert. The end product of this step is a list of concepts which may or may not have any direct relationships. The by-product of this step is synonyms and difficulty levels of the concepts.

The second step in the construction of a CR network is the identification of the direct relationships between the concepts identified in the previous stage. This step is less difficult than the previous step as it is based on the information obtained in the previous step, but it may involve returning to the previous step. In this step, the knowledge engineer puts the concepts along the two dimensions of a large table and questions the expert for possible direct relationships between these concepts. The expert may in this step add, delete, combine or rename previously identified concepts. Deletions may be required by combinations or redundancies. When all the relationships thought possible by the expert are identified, the knowledge engineer puts it all into graphical form for the expert to verify.
may result in going back to the previous step. Once the expert agrees on the graphical representation of the concepts and relationships, the knowledge engineer will ask for the degree of relationship for each of the relationships in the network. This information is recorded and later used in the descriptor of the relationship. The end product of this step is the identification of the direct and explicit relationships between the concepts identified and the degree of these relationships. As a result, a graphical representation of the concepts and relationships is verified by the expert. At any point during this step, we may branch back to the previous step.
The third step in the construction of the CR network is the construction of descriptors. Descriptors play a very important role in retrieval linking and organisation of course material. The key knowledge, or concept definition, which is one of the ingredients of the descriptors is used as one source of evidence in assessing the learner's understanding of concepts. Before construction of the relationship descriptors, the concept descriptors must be constructed. This is because most of the relationship descriptor items can only be built after the corresponding concept descriptors have been constructed. The construction of descriptors is a critical task and requires an experienced subject matter expert for the results to be reliable and obtained in reasonable time. The descriptors contain several information items. Some of this information has already been acquired in the previous steps. These are the names, difficulties and hierarchy levels of concepts and the names and strengths of relationships. The strengths are later modified by the system as a result of a form of parameter learning which takes place when the intelligent assistant is used. The learning component of the system is explained in Chapter 6. Context lines is another item in the concept descriptors. This item is filled during the retrieval of text. Lines from the text which is retrieved are selected and recorded in the descriptor for each concept. These context lines are used as a source of evidence for the learner's understanding of concepts during student modelling. One of the important items in the concept descriptors is types of knowledge. This item is not currently used, but it would be identified in this step if it is used in the future. This item is used for recording the different types of knowledge which will be stored in the knowledge base of the intelligent tutoring system.
For each type of knowledge stored, there must be procedures for manipulating that type of knowledge. This item was added to the concept descriptors to support the use of multiple forms of knowledge in an intelligent tutoring system (refer to Figure 3.1).

The information we need from the expert in this step is the following:

- Sets of keywords for each concept
- Weights of the keywords in each set
- Key knowledge for each concept.

This information is needed for:

- Retrieval of resources
- Automatic hypertext (or hypermedia) generation
- Student Modelling

For each concept, one or more sets of keywords are recorded. It is more likely that there will be more than one set of keywords. Within each set of keywords recorded, a weight out of 100 is given to each of the keywords. They may all be given the same weight, in which case the weight of each keyword is $\frac{100}{n}$, where $n$ is the number of keywords in the set. With each set of keywords the AND operator is implied. The OR operator is implied between different sets of keywords. The NOT operator is explicitly used by putting it before the keyword. Key knowledge contains the main ideas related to a concept. This knowledge is a small subset of the knowledge which may be stored for each concept in the knowledge base. It is
acquired from the expert through interviews using existing knowledge acquisition techniques. Knowledge acquisition in this case is simplified by the fact that each knowledge chunk is acquired with the expert focusing on a single concept. Here, we are only interested in the main ideas and the volume of knowledge is small. The procedure is repeated for all the concepts. For every relationship a descriptor must be constructed. The name of the relationship and the strength of the relationships must be known from the previous steps. The keywords in common are taken from the keywords of the participating concepts in the relationship. The end products of this step are descriptors for all the concepts and relationships. Forms are used for recording this information. These forms contain the information previously acquired and proper empty spaces for each of the items to be acquired in this stage.

The fourth stage in the life cycle of CR networks is testing and verification. In this stage, the CR network is entered into the intelligent assistant. The intelligent assistant provides a facility for entering and maintaining the CR network. It also provides for construction of the descriptors. These facilities will be explained in Chapter 6. The CR model is used for retrieval of text and construction of hypertext from the retrieved text. The results are then reviewed with the expert and inconsistencies are considered. Proper changes to the model are then decided upon. Messages generated by the intelligent assistant during its operation are used for making such changes. The final product of this stage is a corrected and verified model which can be used for preparation of courseware.
The final stage in the life cycle of the CR model is usage and maintenance. In this stage, the system is used for preparation of courseware. The model is customised for such use through the menu options provided. Such customisation is applied only to the customised version but may result in future changes to the model. The original model constructed in knowledge engineering sessions is kept untouched. As the system is used, suggestions for changes to the model are made. The knowledge engineer and the expert consider the suggestions and make the required changes. This may result in transitions to even initial steps in structure elicitation. Such changes may include anything from changes to descriptors to deletion, addition or modification of concept or relationship nodes.

3.7.2 The CR Model for Knowledge Acquisition

The Repertory Grid is a tool used in general Psychology for analysing personal constructs system and the relationships between them. It has also been used as a basis for automation of knowledge acquisition for expert systems. The grid was originally suggested and used by Kelly [1955]. He called it the “role construct repertory test” and used it to study constructs in relation to roles played by people important to a person. In the test, the person is asked to put down the names of people s/he knows. S/he is then presented with the names of three of these people and asked to determine in what important way two of them are alike but different from the third. This type of questioning results in constructs such as “beautiful vs. ugly”, and is continued until several constructs have been elicited. Then the person is asked to put the names of people and the constructs along the side and the
top of the matrix, which forms the grid. The person rates each of the people on each of the constructs. The grid is then analysed to find relationships among the constructs. The result is then checked with the person interviewed for validity. The CR model we have developed can be used as a tool for knowledge acquisition. It is however not an alternative to the grid method but an additional and different tool for aiding the knowledge acquisition process. The CR model can reduce the time spent on acquiring expert knowledge by directing their focus of attention.

Automating the elicitation of the structure of a domain using the CR model, as well as the key knowledge of the domain for the purpose of material preparation, seems to be simpler than knowledge acquisition for expert systems. The reason is twofold. One is related to the structure elicitation and the other is related to the acquisition of the key knowledge.

In structure elicitation we first begin with identification of concepts. The expert is asked to concentrate on the identification of the concepts forming the domain. This step is an important one in constructing a CR model. Once it is done, the expert's focus of attention can easily be drawn to one specific concept for forming the descriptor (including the acquisition of key knowledge) or two concepts for identifying the relationships. Directing the expert's focus of attention to one concept makes it easier to acquire structure specific knowledge related to that concept.

In addition to the knowledge of the domain structure which has to be acquired, we need to acquire the key knowledge of each
concept in constructing a CR model of the domain. This knowledge is minimal knowledge representing only the key ideas of the concept in question. Because of the smaller volume of knowledge acquired, and the fact that the expert focuses on a single concept, the process becomes simpler, making the time spent in this step of the construction of the CR model minimal.

Acquisition of knowledge for building ITSs, as well as expert systems in general, can benefit from the use of a CR model of the domain. In the case of fault diagnosis expert systems, faults can replace concepts. This is why we earlier proposed a two staged approach to knowledge acquisition. In this way the expert can focus on one concept at a time. The use of multiple knowledge representations can be made easier as it becomes a matter local to each concept (see Figure 3. ). The task of knowledge acquisition will remain a critical one, due to human involvement.

Although the knowledge acquisition needs of intelligent tutoring systems are different, the techniques used for this purpose can be applied to other areas. Structure elicitation based on the CR model simplifies the process of knowledge acquisition for courseware development. This is partially because:

1. The expert identifies the concepts first.
2. The expert and the knowledge engineer focus their attention on a single concept and its relationship to other concepts.

The process of structure elicitation is relatively simple to automate. The structure obtained can be used in similar ways to Kelly's grid as a basis for automating domain knowledge
acquisition. The methodology based on the CR model has the following benefits and features:

1. Two-staged approach to knowledge acquisition comprising structure elicitation and domain knowledge acquisition.
2. Simplification of knowledge acquisition yielding greater efficiency.
3. The possibility to include multiple knowledge representations.
4. The attention of the expert can easily be focused on a single concept and its relationships with others.
5. Provision of a domain model, which is important in recent research into building tool generators for knowledge acquisition, as noted by [Eriksson 94].
6. Simple to understand and simple to use.

In constructing a CR model for testing our ideas we selected a limited domain of development. The reasonable number of concepts and the fact that only key knowledge was necessary made the construction of the model possible in the limited time we had.

3.7.3 Factors Affecting the CR Model Construction

Three factors affect the outcome of this process. These in order of importance are:

The Expert
The Knowledge Engineer
Other Sources
The expert is certainly the most important factor. The expert's experience and knowledge, personality and ability to express his/her own knowledge affect the outcome and length of the process with respect to time. These factors are well recognised. What is new here is that the expert must know the structure of the domain and have so-called metaknowledge of the domain. In the case of structure elicitation for courseware construction, the expert must have subject matter knowledge. Not all experts (e.g. teachers) have domain structure knowledge, or even if they do they may not be able to express it. In some cases, it may be necessary to have both an expert teacher and a subject matter expert. Another important factor is the expert's belief in the project. Structure elicitation using the CR model for construction of courseware is something new. Until it is widely accepted or there is a way of showing some results to the expert, s/he may develop doubts about the activities needed to develop the model. The knowledge engineer must be able to attract the expert's attention and belief in the project. Some of the problems experienced in this study during the construction of our academic prototype will be discussed in Chapter 6.

The knowledge engineer's qualities, abilities and familiarity with the methodology are important factors in the success of the project. Qualities a knowledge engineer should possess were discussed earlier. A knowledge engineer in this case should also be familiar with courseware construction, student modelling and be able to understand and communicate with the subject matter expert.
Other determining factors are time and computing facilities, which are needed for testing the model. Retrieval of related material probably requires many hours. The time the expert and the knowledge engineer and the computing facilities are available, and the deadlines of the project, are also important factors which require careful early planning.

3.7.4 Other Potential Uses of the CR Model

We have introduced the CR models for representing domain structure. We have used them to represent the structure of a limited area of geography, for the purposes of courseware preparation. An important by-product well worth mentioning here was the ability to convert text into hypertext automatically. Further, this capability made possible through the CR model can be applied to multimedia generation. Other interesting uses are also possible. Other than the intelligent linking introduced in this work, the CR model may be used as a basis for general knowledge acquisition. Network based hypertext is another place were CR networks can be useful.

3.7.4.1 Network-Based Hypertext

WAIS (World Area Information Servers) is a distributed networked database system which allows its users to do keyword searches for documents at one or several locations (servers). WAIS returns the retrieved documents with a weighted score.
World wide web (WWW) is a hypertext based system which gives its users the capability to navigate through documents over the internet. Documents are marked up using SGML or HTML which are commonly used markup languages.

The use of CR model together with such network based world wide systems can ease the use and maintenance of the hypertext component of such systems. The CR model can also be used for taking advantage of servers meant for production of courseware. Network based applications will not be discussed in this study.
4. HYPERTEXT AND CONCEPT RELATIONSHIP NETWORKS

4.1 General

The word hypertext dates back to the early 1960's [Yankelovich, 88]. The ideas, however, existed two decades before the word hypertext was coined by Ted Nelson. Hypertext refers to a system of linked text with paths connecting related information. The term hypermedia is used when other media are involved. In a hypertext system, text may be combined with databases, commercial information, graphics, images, or spreadsheets [Gillman, 88]. Hypertext techniques provide powerful tools for technical, commercial, management and educational applications. Hypertext systems are used in different ways in different environments. Hypertext is considered to be theoretically and operationally consistent with educational design [Jonassen, 92]. Our study is focused on educational applications, and a model called the CR Model has been proposed. However, the model proposed, and the techniques used, are also applicable to general hypertext systems.

A hypertext system, as indicated above, is a collection of documents which are linked together. The links may be a part of the document or kept separately, as depicted in Figure 4.1. A user can navigate through related documents by following the links. This may, however, cause the user to get lost and lose orientation if the
Figure 4.1 Links in Hypertext

a. As Part of the Document

b. Links Kept Separate (as a Web)

volume of text is large. Disorientation is a known problem of hypertext systems. This problem has been dealt with by others by:

1. Displaying several windows on a single screen
2. Providing for quick response when the user is following links to enable quick re-orientation.

Another problem of hypertext systems is the high cost of the repeated operation of creating the links. Automatic linking is the solution to this problem. The CR Model proposed in this study solves these two major problems of current hypertext systems by providing an intelligent higher level layer of links above the document level and making automatic linking possible.

Usually at any given time the user of a hypertext system is interested in only a portion of the information. This is especially so in the case of the education area with which this study is concerned. The system constructed using the CR Model allows the user to select a small portion of a network which is customised according to the user's needs. This overcomes the problem of hypertext systems known as information overload.

Hypertext documents do not have a specific size, even though some systems designers have used a certain limit for their document size. In principle a hypertext document may be a paragraph, a section in a book, or even an entire book. Different orders of presentation can be imposed on a document by the system depending on the user of the system [Easterbrook, 89]. This ordering is clearly non-linear and hypertext is sometimes referred to as non-linear text. It is possible to use graphics to show the links or to show several documents on the screen, making it easier to navigate through the network of linked documents. Some browsing systems, often those based on laser-disk technology, are
static and do not allow the user to create new links. Some systems do allow the user to create new links and to add comments to documents or to connect comments to documents. Some systems allow for multiple users to access and create links among documents. Many of the issues of concurrent databases become applicable to hypertext systems.

4.2 History of Hypertext

Vannevar Bush in his 1945 paper titled, "As We May Think" proposed the idea of a hypertext system and called it a Memex.

![Diagram of Bush's Idea of a Network of Linked Documents (Memex)](image-url)

**Figure 4.2 Bush's Idea of a Network of Linked Documents (Memex)**
Memex was never implemented on a computer, perhaps because the necessary computer was not available in those days. Bush wanted to use microfilm and photocells to implement Memex. Figure 4.2 [Easterbrook, 89] depicts Bush's idea using a network of connected items. Bush called for a mechanisation of the scientific literature and introduced a machine for browsing. Due to the lack of the required computing power for implementing Bush's Memex, he used microfilms and photocells. Bush writes: "The human mind operates by association. Man cannot hope fully to duplicate this mental process artificially, but he certainly ought to learn from it. One cannot hope to equal the speed and flexibility with which the mind follows an associative trail but it should be possible to beat the mind decisively in regard to the performance and clarity of the items resurrected from storage" [Bush, 45]. In Memex, to associate two documents, the user enters a link name in a code space at the bottom of the viewer which also contains photocell readable dot code naming the other document. By tapping a button under the code space the other document can be viewed.

The term hypertext was not used before Nelson mentioned it. The first system called a hypertext system was Nelson's Xanadu, built with a long term aim of putting the entire world's literature on-line (an on-line library).

About twenty years later, Engelbart of Stanford Research Institute, building on Bush's ideas, proposed a system which could be used to store specifications, plans, designs, programs, documentation reports, bibliography and reference notes to allow a project team to do much of their intercommunications via
consoles. A few years later this system was implemented and called NLS (oN Line System). NLS's successor, called Augment, is marketed as a commercial hypertext product for project information management and software engineering. In recent years several other hypertext systems have been constructed as both commercial and research prototypes. Some of the most important Hypertext which focus on education will be considered here. Other hypertext systems such as Neptune [Delisle, Schwartz, 86] Augment and Shadow [Carando, 89] which were meant to be hypertext tools for software engineering will not be discussed. The principles are, however, the same. Another idea related to this work is the idea of the electronic book which has been discussed as having potential uses in education. The idea of electronic books will be discussed in the following section, but we will not consider the details of any of the many existing electronic books.

4.3 Hypertext Models

Although other models exist, the first, most frequently referenced, but not so frequently used model is the "Dexter hypertext model" [Halasz, 89]. The Dexter model tries to explain the existing hypertext systems and to be a model for future hypertexts. Components (nodes) can be atomic or composite. Nodes are connected via links. The end points of links are called anchors. An atomic component consists of three sub-components: content (text, graphics, etc.), attributes (description of the component) and presentation specifications. A composite component is a collection of components. Links in this context are not part of the documents.
Three layers are assumed for a hypertext in the Dexter model. The details of the contents and the internal structure of the components are stored in a detailed layer called the 'within-component layer'. The structure of the hypertext is stored in a 'storage layer'. This layer is the essence of the hypertext as it is the actual network of nodes and links. A 'runtime layer' stores the presentation specifications.

A hypertext system can be represented as a graph, with each document represented as a node and links represented as the arcs of the graph. Graphs have been used to study the structure of hypertexts [Hara, 91]. Petri Nets [Peterson, 81] are an effective tool for describing and analysing flow of control. Petri Nets are representable as directed graphs. They are automatons and have formal language properties which makes them language generator/recognisers as well as abstract machines.

Another formal hypertext model is the Trellis model [Stotts, 89]. Based on Petri Nets, Trellis is a generalisation of the existing graph based hypertext. Trellis uses Petri Nets as a basis. In Trellis, structure and contents are separated. In a hypertext system based on Trellis, each node is a Petri Net place and each arc is a transition. In Trellis, a hypertext consists of a Petri Net, a set of components (contents, windows, buttons) and a set of mappings between these components, the net and the display. More formally, a hypertext in Trellis [Stotts, 89] terms is a six tuple,

\[ H = <N,C,W,B,PI,Pd> \text{ where,} \]
\[ N = <ST,F> \text{ is a Petri Net;} \]
C is a set of documents;  
W is a set of windows;  
B is a set of buttons;  
PI is a logical projection and  
Pd is a display projection for the document.

Each hypertext uses its own model and there is no widely accepted standard. Of the two models explained above, the Dexter model is more frequently cited. However, because each no standard has been agreed on, the models are not widely used although many existing hypertexts are close to Dexter's. The CR model proposed in this study is closer to Dexter reference model.

4.4 Automatic Linking

4.4.1 Overview

The bottleneck in creating hypertext systems is the expensive and tedious task of link creation. What we have said so far and most of the work done to date assumes that the links are created manually. The little work done on automatic linking is based on the structure of the content which is not very useful. The few limited attempts at automatic semantic link creation have not been very successful. In the following paragraphs some of that work will be discussed briefly.

Before any hypertext creation is attempted, the text must be in electronic form. The text used in this study was geography text, scanned, as is commonly done, with a scanner using OCR software.
Most text today is available electronically so this aspect is not an issue in this study.

Although automatic hypertext linking has being investigated by several researchers, the achievements have been modest. There is not a standard or effective method of automatic hypertext linking. The work done to date can be categorised into two approaches.

The first uses the structure and layout of the document. The structure of the document is either identified using the document layout [Fujisawa et. al., 92] or a markup language is used. The most widely referenced markup language is SGML [ISO 8879]. Most documents created by today's wordprocessors contain markups (bold, etc.) which are created for the purpose of typesetting. These markups are called procedural markups. Descriptive markups mark certain words or sentences. In both cases tags are used. Each wordprocessor, however, uses its own markup tags [O'Conner, 92]. A standard such as SGML would be very effective if adopted more widely. Tagging the documents can be done manually or automatically using the layout of the document. Most of the work done on automatic hypertext linking [Plaisant, 89; Fujisawa et. al., 92; Furtta et. al., 90, Futura et. al. 89] falls under this approach. The problem with this approach is that the links are structural, and not referential, which makes their use very limited.

The second category uses statistical and/or knowledge based techniques, using Lenat's CYC knowledge base [Lenat 86; Lenat, 89] for instance, for automatic linking [Lelu, 91; Clitherow 89; Hays
These studies address the problems of information retrieval. They are currently of limited use. Other work such as that of Gloor [Gloor, 91], although good for aiding navigation, is aimed at generation of overview diagrams rather than automatic linking.

4.4.2 CR Model for Automatic Linking

The CR Model explained in Chapter 2 has been used for automatic generation of semantic links among documents in the geography domain. The CR Model provides an intelligent higher layer of links above the document level, which is created through interviews with an expert using the knowledge engineering techniques explained in Chapter 2. One advantage of having such a higher layer is that it solves the problem of disorientation in hypertext systems. The main advantage of the CR Model is that it provides for automatic associative linking of documents. At any time the user is interested in a certain portion of the text. For instance, in a diagnosis and repair environment (replacing concepts with faults), the user is specifically interested in material related to the particular fault, or, at most, other related faults. A similar situation exists for concepts in an educational setting.

Sets of keywords defining each concept are used as a basis for retrieval of information. The automatic linking takes place while the text is being retrieved. As the text is retrieved the keywords used for retrieving the it are checked against the keywords defining other concepts. Synonyms are also used in this process, which results in links from each concept to the text of other concepts which are sometimes not directly related or where
there is no prior information about them being related. This information can be used to suggest links at a higher level of the CR network. Links are also created on the basis of relationships existing in the CR network. The text being retrieved is indexed by the system before any retrieval takes place. It does not, however, have to be in a specific format. Text analysis [Jones, 90] is not necessary, although it may be a good area for research for better performance in this context. The text base may be added to or deleted from without any risk, or it may be totally removed and replaced with a different one. As new material is published, the text base kept current. As the field changes, so can the CR network and the information in the descriptors of the concepts and the relationships. The system provides a mechanism for these changes, and makes suggestions to aid in the process, using a form of parameter learning explained in Chapter 3 (Information Retrieval).

4.5 Electronic Books and Encyclopedias

4.5.1 An Overview

Several dynamic structures, sometimes called electronic books, have been reported in the literature. Among these are Book C [Weyer, 82] which was developed as a result of Weyer's Ph.D. studies, A Prototype Electronic Encyclopedia [Weyer, Borning, 85] and a Hypertext Medical Handbook [Frisse, 88].

There have been attempts to formalise the knowledge embedded in text for problem solving in electronic books and encyclopedias [Lenat et al., 83; Harding, Quinney, 90]. Electronic
encyclopedias are certainly valuable educational tools, especially if they are accompanied with problem solving knowledge. We will, however, concentrate on general hypertext systems as the underlying ideas are very similar.

4.5.2 Intelligent Tutoring Systems and Dynamic Books

The sequence in which material is presented to the student in an ITS is usually determined from a student model using a plan. The student model is used to determine the student's state of domain knowledge. This in turn determines what is to be taught, or not taught, and what teaching strategy is to be adopted. In conventional computer assisted instruction systems, the sequence is usually determined using branching techniques. Another possibility is that the students be allowed to decide on the sequence of material that is to be presented to them, which will probably be different from a sequence generated by a teacher [Benest, 90]. An effective sequence will often depend on the student's state of knowledge, and not many students can generate a sequence of material to be presented. A sequence of material to be presented can also be generated automatically using an initial student model, if one exists, and a set of facts defining the prerequisites of various topics or a lesson hierarchy.

A book has a structure which allows a student to select what pages to read first and what sequence of chapters to follow. Text books often have a number of questions at the back of each chapter and the solutions at the back of the book. A book has physical limits. Pages are limited and numbered and the reader has a feeling
of where in the book s/he is. This is not true, however, in an electronic book or encyclopedia. These and other features of a paper book such as the possibility to mark certain parts of a book or to write comments in the margins and the use of place holders can be implemented in electronic books. A manual book cannot animate or simulate the actions or experiments it is explaining. It can, however, be argued that the use of numbered figures and diagrams is a way of overcoming this shortcoming. Even so, because many details are left out when an main issue is presented in a diagram, a diagram or a set of numbered figures cannot be as effective as animations and simulations. An electronic book can provide such facilities, which is why they are sometimes called dynamic books.

A manual book has an index at the back, in which users may locate their desired subject. In an electronic book, the index is in the form of hypertext links allowing the user to navigate through the book exploring desired topics so that almost all the functionality of a manual book can be implemented in an electronic book, as well as the facilities mentioned above which are possible only using an electronic computer. Electronic books have worked well when used by students for finding answers to questions assigned them [Weyer, 82], but existing electronic books or manual books cannot answer questions or solve problems. This is clearly because problem solving knowledge is required to do this. As discussed earlier (in Chapter 1) a variety of knowledge and in turn representation formalisms (including text as on form) must be combined in an authoring shell to be able to teach using a machine.
There are attempts to combine knowledge representation and hypertext techniques to bridge the gap between intelligent tutoring systems such as GUIDON [Clancey, 79] and the recent electronic books and hypertext systems. An example of such work is the attempt by Lenat et al. [90] who have reported work underway at Atari, Inc. for constructing expert systems using encyclopedic knowledge. KNOESPHERE, once completed will consist of a large collection of frames formalising the paragraphs of an encyclopedia. The work of [Harding and Quinney, 90] is another example in the limited domain of numerical analysis. Trying to fill the gap between the current ITS and hypertext technologies is in fact what this thesis will contribute to.

In this section, electronic books and encyclopedias were briefly overviewed. The underlying idea of electronic books is very similar to that of hypertext systems. Thus, in the following section, only some existing hypertext systems will be discussed, when we survey a limited number of existing hypertext systems which share many features in common with the electronic books and encyclopedias we have just discussed.

4.6 Some Existing Hypertext Systems

4.6.1 Textnet

Textnet is a hypertext system well worth mentioning here, as it is the result of the first Ph.D. thesis on hypertext and the predecessor of Xerox NoteCards. Randall Trigg the principal architect of NoteCards [Conklin, 87] completed his Ph.D. at the
University of Maryland in 1983. He proposed over 80 types of links that may be used for linking documents and comments to documents by collaborators and critics. Among these link types are refutation and support links as well as links such as point irrelevant and data inadequate. In Textnet, paths may be defined through text.

What is interesting in Trigg's thesis is his view of text handling and the hypertext technology:

"In our view, the logical and inevitable result will be the transfer of such activities to the computer, transforming communication within the scientific community. All paper writing, critiquing and refereeing will be performed on line. Rather than having to track down little-known proceedings, journals or unpublished technical reports from distant universities, users will find them stored in one large distributed computerised national paper network. New papers will be written using the network, often collaborated on by multiple authors, and submitted to on line electronic journals" [Trigg, 83]. In the view of the author, there should also be collections of educational material (books, papers, photograph images, etc.) stored on mass storage devices and made available through networks to educational institutions. Having such a facility available, one could use the material, which could be updated frequently, to construct courses by retrieving related material and organising it using hypertext and ITS techniques. This organisation can automatically be done using a concept relationship (CR) network described in this thesis. In fact, telecommunication organisations could be the providers of such hypermedia services at a cost.
A typical scenario would be as follows: A teacher views a CR network after entering the topic s/he is interested in. S/he then selects the desired portion of the network, and verifies or modifies the relationships among the concepts as well as the degree of relationship. The system then retrieves related material for each concept and connects it using hypertext techniques. Once this process is complete a custom CR network can be constructed. Students can interact with the CR network or browse through the retrieved material. While the student is trying to understand the inter-relationships among the concepts, a model of that particular student is constructed. Using the retrieved material the teacher can use an authoring shell to construct lessons. The student models are used as initial student models in the authoring shell and can be used as a basis for intelligent linking at document level as explained in later sections.

Details of Textnet will not be considered here as similar ideas were also used in NoteCards which will be explained next.

4.6.2 NoteCards

The main architect of NoteCards is Trigg who was mentioned in the previous section as the first person to do a Ph.D. on hypertext. NoteCards was developed at Xerox PARC on their D-Series LISP machines. NoteCards has been built on the top of several Xerox LISP packages. It provides for organisation, management and display of interconnected NoteCards. Figure 4.3 [Conklin, 87] shows a screen from NoteCards. Large databases
Figure 4.3 A Typical Screen from NoteCards Showing FileBox and Text Cards

consisting of up to 1600 nodes with 3500 links between them have been constructed using the NoteCards [Conklin, 87]. Halasz et al. [Halasz et al., 87] report NoteCards being used in what they call idea processing applications (e.g. writing research papers, and design) by many people. NoteCards is similar to Augment [Carando, 89] in the goals both are trying to achieve.

The idea in NoteCards is to have a collection of related information in the form of text, graphics and images residing on
so-called NoteCards connected via links. Links are of different types including support, augment and comment which are very similar to those used in Trigg's Textnet.

A notecard resembles the electronic form of a paper notecard. It contains editable material of various types, including text, drawings, and bitmap images. A text card contains editable text and can be retrieved through its contents, and displayed on the screen using standard Xerox Lisp windows, as shown in Figure 4.3. An image card contains a bitmap image. A user may create his own card type, differentiated by the nature of its contents - for example an animation card.

There are two special types of cards used to manage other cards and the links between them. The first is the Browser card, which is used to show the network of various types of cards. Each notecard is represented by a box containing the title of the card. Links are represented as lines. A user can edit the structure of the network using the Browser cards. The second is a Filebox card, which is used to organise or categorise collections of notecards. Each notecard must appear in at least one Filebox card, while Filebox cards can appear in other Filebox cards. The hierarchical structure enables users to maintain large collections of material, independent of their network connections.
4.6.3 InterMedia

InterMedia is the result of two years of work by a large group of systems developers and researchers at Brown University. InterMedia is meant to be used as a tool by both authors and readers. It has been used by professors to organise course related material, and by students to read the material and add their comments. It runs on a network of Unix workstations. A text editor, a graphics editor, a three dimensional object viewer, an image viewer and a timeline editor are integrated to provide a multi user environment for creating and editing linked material and navigating through the links. Similar to Macintosh, a window manager and a graphical folder system is used. InterMedia is built on twenty years of experience at Brown University on three generations of hypertext systems [Conklin, 87], namely,

Hypertext Editing System,
File Retrieval and Editing System,
Electronic Document System.

InterMedia has been used at Brown University for educational purposes. Several courses in humanities and natural sciences have been taught using it [Nielsen, 90], including one in cell biology and another in English literature. Students in Bio106: Cell Biology in Context at Brown University use InterMedia to explore material about cell biology and to link them to write their term papers. The
user of Intermedia's cell biology material has to select and open an icon titled "Bio 106" [Yankelowich, 88]. This opens a web connecting a number of documents. Some of these are images, some are text and some are graphics. The user can choose to open any of these
documents to inspect or to edit them (see Figure 4.4 [Conklin, 87]). Webs are similar to FileBoxes in NoteCards.

Intermedia allows for links from a location within one document to a specific location in another document. When a link is created, properties such as user I.D. and creation time are assigned to it. Unlike those of NoteCards, InterMedia links are not a part of the document. Instead a web is used to store link information. A web is accessed by its creator and those having access rights to it. Users working in a particular context open the corresponding web which imposes its links on the documents. So, each group of users will use their own web for navigating through the documents. A protection mechanism checks access rights to documents and webs.

InterMedia uses an architecture based on the object oriented approach and building blocks, which are sets of reusable classes used for implementing basic functions. Both the object oriented approach and the concept of building blocks provided a methodology and a development environment for the large team of developers working on the project.

Yankelovich et al. [88] report that InterMedia was used by 8 authors and 80 students. Approximately 850 English related and 200 biology related documents were used and 3000 links were created among them.
4.6.4 KMS

ZOG, the non-commercial version of KMS, was developed at CMU in 1972. ZOG is a multi-user hypertext system which aims at constructing large databases of small documents in units called frames. Developers of ZOG started a company called Knowledge Systems and developed a commercial version of ZOG and called it KMS (knowledge Management System). A command line called the global pad appears at the bottom of a frame as shown in Figure 4.5. Some of these commands are:

- Comment
- Edit
- return
- help
- Save
- Exit
- Back
- Next
- Prev
- Reset
- Mark
- Goto
- Info.

KMS has been used in different applications as an information management and as a CAI tool. The most interesting application of KMS was its use as an information management system on board the nuclear powered USS CARL VINSON aircraft carrier. This version of ZOG/KMS was used for:

- Online policy manual
- Interactive task management
- Online maintenance manual
Developers of hypermedia systems face many design issues. The design of KMS, a large-scale hypermedia system for collaborative work, ...

Introduction
1. Background
2. Overview of KMS
3. Hypermedia design issues
4. Conclusion
Acknowledgements
References

@Old Version
@Proposal to restructure
@Make backup tape for interest

Save Exit Reset Prev Next Home Goto Info Disp ...

Figure 4.5 A KMS Frame

Interface to the Airplan expert system

At Westinghouse KMS was used for providing rapid access to emergency operating procedures for nuclear power plant operators.

KMS shows a frame at a time to the user. It can be criticised for causing user disorientation because of this. Users viewing one frame at a time are likely to lose orientation. KMS, however, minimises response time to overcome this problem. Users can navigate from frame to frame and regain orientation very quickly.
A frame may contain text, graphics and images. A frame is shown in two windows side by side on the screen. If a frame contains complex diagrams it is shown as a full screen. Users can navigate through frames, edit frames or add new frames. Adding a new frame is very similar to navigation to an existing frame. A user can invoke programs as well as using the commands provided on the command line. A language like Apple's Hypertalk is available to users of KMS.

A typical frame of KMS is shown in Figure 4.5. A frame title describes the topic of the frame. The frame name which is unique to each frame also appears on each frame. The command line provides for different operations to be performed at user's request. Tree and annotation items are used for linking frames. Some hypertext systems (e.g. NoteCards, Textnet) allow user defined links and some only allow predefined links. A tree item link in KMS is used for connecting a frame to frames in a lower level of hierarchy. In other words, they implement structural relationships. Annotation items begin with '@' and are used for implementing associative relationships, including comments and cross references. In KMS, unlike NoteCards and InterMedia, only one type of node is supported, which is a general frame used for text, graphics and images. NoteCards Fileboxes provide for a graphical view of the underlying database structure. No such graphical browsing is supported in KMS. The goal of KMS is to make construction of large databases more convenient by providing rapid creation, editing and navigation through frames.
4.7 Using Hypertext Techniques in Education

The aim of our study is not to construct a hypertext system like NoteCards or InterMedia, even we will produce a hypertext based system. Hypertext will be the experimental medium for application in Intelligent Computer Assisted Instruction. We have constructed a tool that can be used as an aid for producing instructional material. Teaching is a task which requires expertise, and the purpose of intelligent tutoring systems is to use artificial intelligence techniques in the construction of educational software. The approach proposed in this study is to construct an intelligent network of concepts and relationships (explained in Chapter 2) which I call a concept relationship network (CR network) on the top of a network of hypertext documents. Hypertext documents linked through a CR network for a subject and a lesson hierarchy is the way to assemble a lesson for use in an authoring shell environment. The intelligent assistant constructed for this purpose can also be used for retrieving and putting together pages of an operation or repair manual which can then be used as an explanation front end for a diagnostic expert system. Such a facility, used with an expert system, can aid the technician/operator to retrieve repair/operation procedures or instructions when problems occur or faults are located.

4.7.1 Architecture of the Hypertext Component

The main building blocks of the system are a concept relationship network, a knowledge base, stacks containing hypertext documents, a retrieval component and a user interface,
as shown in Figure 4.6. Initially, a large collection of documents resides on a mass storage device such as a CD ROM. The concept relationship network acts as an intelligent higher layer to the stacks containing related documents. The concept relationship network is used for retrieving related documents for each of the documents which are organised in stacks. Each concept has a corresponding stack of hypertext documents. These documents can be displayed and edited by the users.

The retrieval component is driven by the CR network, and a detailed explanation of precisely how is given in Chapter 6. The knowledge base contains rules and facts defining the concepts. The knowledge will be formalised in the language of Logic Manager [LM, 91], an Apple product used in the implementation of this system.

The user interface is a mouse and menu based interface. Three types of cards are used, one for displaying the CR network, another for displaying hypertext documents, and the third for aiding in the navigation. The CR network constructed through knowledge engineering sessions is displayed on cards. The user can explore the network which will usually reside on several cards. A typical CR card is shown in Figure 4.7. The user can hold the mouse down on a concept node to choose from a pop-up menu. An option of the menu allows the user to browse through the stack associated with that node. Stacks are constructed using the retrieval component of the system. Holding down the mouse button on a concept node causes a pop-up menu to appear, giving the user various options for modifying the network or getting information and modifying the node descriptors associated
Figure 4.6 The Hypertext Subsystem

with the concept nodes. The user can navigate through the network itself and explore it using the mouse.

Hypertext stacks are collections of cards containing hypertext documents related to their corresponding concepts. This relationship is depicted in Figure 4.8, which also illustrates the intelligent layer constructed on the top of the hypertext links. Documents are displayed in scrolling fields on text cards on the
Figure 4.7. An Example of a CR Card

screen. As shown in Figure 4.9, a text card has five buttons. The user can always click on button "CR network" to go back to the network. The concept of a text card will be explained in the next section.

In NoteCards, nodes and links are not kept separate; links are a part of their source node. In InterMedia, links are represented separately from nodes in the form of webs. As explained earlier, in our approach, our links at the document level are similar to those of NoteCards. However, there are intelligent links at the higher concept level in the form of concept and relationship nodes. Thus our system is different from both InterMedia and NoteCards, and
Figure 4.8 Hypertext Document Stacks and their relationship with the Concept Relationship Network

any other existing hypertext system. The relationship nodes link groups of hypertext documents related to concepts. In other words, there are two levels of links, and the links at the higher level are links which are named, and defined, as explained in Chapter 2. The higher level structure can be organised into a lesson hierarchy if users wish to use the hierarchy levels associated with each concept or choose their own. The level of relationship among the concepts is used for filtering out less relevant concepts. Retrieved documents can also be filtered, since the user can edit, add comments, add his own new documents, or exclude unwanted documents.
good reason to suppose that malnutrition, in one form or another, is the most serious problem now facing the human race. On the one hand, an affluent few are inviting arteriosclerosis (an accumulation of hardened fatty deposits in the arteries) by consuming too much food rich in animal fats—a steady diet of fatmeat hamburgers, ham-and-mayonnaise sandwiches, ice cream, and the like. On the other hand, perhaps as many as two billion people are prey to infection and deficiency diseases because their diet does not give them the needed minimum of proteins, minerals, and vitamins. Among the latter, it is the children from one to five years old who suffer most.

Deficiency diseases

Where hunger and malnutrition exist, an unusually high percentage of babies born alive have a low birth-weight and incidence of prematurity is higher. It is also considered that in any country a high mortality rate in the one-to-four years age group indicates widespread malnutrition. While the immediate cause of death may not be a nutritional (deficiency) disease, malnutrition lowers the resistance to infectious diseases and the ability to recover. Even

Figure 4.9 An Example of a Text Card

4.7.2 Operation of the Hypertext Component

The user views a CR network and the system allows him to select the portion he is interested in. Once the selection is made, a customised CR network containing only the selected portion of the network, with the changes made, is created. The selection is done in one of three ways. Either the network as a whole, or individual concepts on a one by one basis, can be selected. The user can choose one node and let the system select the portion of the network related to the selected node automatically, in which case
the user can set the strength of relationship (explained in Chapter 3) to be used as a basis for the selection. Each node has a descriptor containing various types of information, both acquired from the expert and generated by the system associated with it. Among the fields of the descriptor of each node are sets of keywords defining each concept. Sets of keywords are acquired from the expert in knowledge engineering sessions. Keywords, along with synonyms, are used for retrieval of text through the retrieval component. The text retrieved is linked to its concept node in the CR network. The text is locally connected so that a user can click on words and navigate locally through the text of a concept which is interrelated. In addition to the links at the CR network level, links are also created among the text of concepts at the document level. These are called external links. A user can navigate through the hypertext external to the text of the concept he is currently exploring by clicking on a button 'external' (Figure 4.9). At any given time, the user knows what concept he is exploring and an instant jump to the corresponding node in the CR network is possible by clicking on button 'CR network'. The user can also go back to the text of the previous concept without going up to the network layer by clicking on button 'Back'. The external links are created while the retrieval component is retrieving text based on the keywords defining the concepts, the synonyms, and the relationships among the concepts.

The hypertext component allows the user to explore the text. This is exploratory learning, but guided learning is also possible using the notion of intelligent links at document level introduced in this study. The system uses three types of cards, namely, text
cards, network cards and external navigation cards. A text card, as shown in Figure 4.9, consists of a scrolling field containing a piece of related text and a collection of buttons. A word in the text or the buttons may be clicked. Clicking on the button 'External' will set the navigation mode to external, causing only the external links to be explored when a word is clicked. Clicking on a word while external is set to off causes the navigation to be limited to links internal to the current concept. Clicking on button 'Concept' will allow the user to go to any of the concepts in the CR network.

In external navigation mode, when the user clicks on a word, an external navigation card (Figure 4.10) is shown, displaying concepts related the clicked word. The user can choose between the concepts. Clicking on a concept will result in navigation to the text related to that concept. A following internal navigation causes the related text card to be displayed. All the linking is done automatically, which is a considerable saving in hypertext development time.

4.7.3 Advantages of the CR Network Approach

The main advantages of the CR network in hypertext are:

- Solving the problem of disorientation
- Automatic Linking,

both of which were explained earlier in this chapter. The user may not know whether she wants to pursue a link as it may not be related. The advantage of the CR network is that all links created locally to a concept are related, and the concepts themselves are related to a known degree and type of relationship.
In a hypertext system, a node may reside in the memory, on disk, on CD ROM or somewhere else. The user doesn't know what kind of a delay to expect when following links of the network. Using a CR network, documents are retrieved and links are created based on the concepts and the relationships defining the topic of interest. More specifically, in this implementation, once the documents are retrieved, they are put into Hypercard stacks and accessed from there, which also solves the problem of information overload.

![Diagram of navigation card]

**Figure 4.10 An Example of an External navigation Card**

The type of learning usually supported by hypertext is discovery learning, where the students explore what they are interested in. Using CR networks themselves, and the student
model generated using it, intelligent links may be created to provide for guidance for the student exploring the hypertext. As explained in Chapter 6, intelligent links at document level are possible and were introduced in this study to provide for such guidance.

Other advantages of this approach not directly related to hypertext systems are discussed under similar sections in related chapters.

4.7.4 Learning Through CR Networks

In this chapter, hypertext technology was studied and several existing hypertext systems were discussed. A hypertext system based on the idea of a concept relationship network using hypertext techniques was proposed and explained. Both hypertext systems [Gay, Mazur, 91] and electronic books [Weyer, 82] seem to have worked well when used by students for finding answers to questions assigned them. Hypertext is consistent with educational design [Jonassen, 92]. It can well be used for exploratory learning. The document level intelligent links introduced in this study, in conjunction with the student model constructed using the CR network, can make guided learning with hypertext possible.

In summary, the contribution of this work is that hypertext links have a higher level intelligent layer which links documents of related concepts together. This not only solves several well known problems of hypertext, but makes automatic linking easy. The notion of intelligent links makes hypertext more suitable for
educational environments, although we are using it as a tool for intelligent tutoring systems. Both the CR networks and the hypertext documents can be explored by the user. They are both used for building a model of the student using the system. The information gathered from interactions with the student is used for constructing an initial student model, as we discuss in the next chapter.
5. STUDENT MODELLING

5.1 General

"User modelling" is the general term used for a body of techniques for individualising software systems. The way the system treats its user is altered according to the user's characteristics or behaviour while interacting with the system. Even though modelling users in this way is not straightforward, doing it will improve the performance of the system. The topic of user modelling has recently attracted more attention. Examples of such work are a study of the effect of cognitive style on learning [Coventry, 89] and collecting knowledge about users in intelligent interfaces [Jerrams-Smith 89]. Fisher [1989], Scherz et al.[90], French [90], Sleeman [87], Brown and VaneLehn [80] and Cuff [1980] are among other references addressing the issue. User modelling may be explicit, putting most of the responsibility on the user, or implicit, and inferred by the system. Explicit modelling of the user can be done by asking specific questions or by letting the user choose from a set of possible alternatives. The selections are kept in a file and used to manage the interactions of the system with the user. In implicit modelling of the user, parameters related to the user's characteristics (eg background, sex) and
behaviour (eg problem solving ability, pace) must be considered. Depending on the type of the environment, and the system, the user model may be a model of a canonical user or several models may be kept, and user characteristics matched against the models at hand. A third possibility is to infer a model, or a combination of models. If the inferred models are kept and the number of users is large, then there may be a problem of space for saving the models. Despite this, it may be beneficial to do so. The time needed for construction of the model is also important and should not be long enough to affect the overall performance of the system. [Rich 85] recognises three dimensions for the universe of user models as being:

1. a single user model embedded in the system, in contrast to individual user models constructed as a result of interactions with users.

2. explicit models, in contrast to models inferred by the system from the user's behaviour.

3. models based on user characteristics, such as background knowledge, expertise, and areas of interest, in contrast to the current behaviour of the user while solving a problem.

Depending on the system and the environment, the type of model required must be identified, and the proper actions taken. Note that there are different ways of modelling a user, depending on the objectives. As discussed earlier, the model could be a single canonical model, several general user models, frames into which
a user may fit, or a model constructed for each user by the system. In the latter two cases, conflicts may take place both in the model that the current user fits into, and the information collected about the user at different times. It is the responsibility of the system to resolve these conflicts. In some cases the model may not be an appropriate one at some point in time, and a new model has to be constructed.

Long term studies and research are necessary to discover methods which can lead to the construction of user models which substantially improve the performance of software systems. One area of computing which has attracted most of the attention in user modelling is ICAI. However, researchers are still concerned with user modelling in general terms as well [Rich, 85; Sleeman, 85]. In the following paragraphs, student modelling will be overviewed, and different methods of constructing and maintaining such models will be discussed.

In an ICAI system, constructing a model of the learner is an issue of major concern for individualising instruction, and it will be discussed in the following section. In an ICAI environment, since the user is most often a student, the term student modelling is used instead of user modelling, which is the more general term. A student model may be very simple, in the form of assertions about what the student knows, or it could be very sophisticated, containing information about what the student knows, doesn't know, or knows incorrectly, and even hints about the cognitive state of the student. Several methods of modelling the student have been developed by various researchers in ICAI. Simple and ad hoc models [
Peachey and McCalla, 86; Heins and O'Shea, 85; Gable and Page, 80] will be discussed in Section 5.3, followed by a discussion of more sophisticated student modelling methods [Clancy, 79; Clancy 87, Sleeman and Brown, 82; Sleeman, 85; Wenger, 88]. Overlay models will be emphasised in this study. Perturbation models and differential models based on student misconceptions will also be discussed.

5.2 Individualised Instruction Through Student Modelling

For years, people involved in instruction have been thinking that computers will someday fulfil the promises of individualised instruction. Conventional CAI has not been able to individualise instruction. It can only provide a one-to-one physical instructional approach (a computer or terminal per student), in which the material presented and the way of presentation are the same for each student. In other words, the material is not tailored to fit the needs of an individual student (background, interests, etc.).

One should not confuse one-to-one (individual) instruction with individualised instruction. Individualised instruction doesn't necessarily mean one teacher per student. It could take place in a group, as long as the material is presented in a way that takes into account the peculiarities of each learner. In such a case, the order, pace and other specifications of the material presented are important. In this context, a course may be presented on a time shared basis to different students, taking each student into account. Clearly this is very difficult for a human teacher to manage, especially if student numbers are large. For a computer program to
do such a thing, it must construct and maintain a model of the student during the course of instruction. As suggested by Yang [1987], individualised instruction should provide for alternative contents and instruction approaches, and data concerning individual student's background, interests, abilities, learning style, academic records and requirements set by the administration, should be taken into consideration.

As stated earlier, most conventional CAI systems do not provide for individualised instruction, but rather one-to-one instruction. Artificial intelligence has been considering this problem for many years and systems employing AI techniques for modelling the learner have been somewhat successful in this context. Two approaches to modelling the learner from the point of view of this work are overlay models, and models based on student misconceptions, which will be discussed later.

Yang [1987] suggests some parameters for individualised instruction, each of which is controllable by an ideal student model if such a model can be constructed. The current technology is unable to produce such an ideal model but we are not far from achieving the goal of individualised instruction using student models. The parameters suggested by Yang are: instructional amount, display time, instructional sequence, personal attention and internal learning activities. Instructional amount is the amount of instruction a student receives according to her ability. Display time is the pace at which displays are presented, which may be controlled by the user at press of a key or set by the system for each user. Instructional sequence refers to the sequence in which
the material is presented, which must be altered to fit the specifics of a particular learner. Personal attention is the reinforcement a student receives, and, finally, internal learning activities refers to the use of external information from the student to interact with her internal learning activities.

5.3 Simple and Ad Hoc Methods for Constructing a Student Model

A tutor who knows what the student knows incorrectly, or student misconceptions, can tutor better than one which does not. This is, however, not an easy task. For each domain, experts would have to work for long periods of time to predict misconceptions which may occur in that particular domain. For a simple domain such as arithmetic operations [Burton and Brown, 78], or even simpler, the domain of subtraction [Young and O'Shea, 81], an unbelievably large number of misconceptions are possible. It is specially impractical in an authoring shell environment unless the misconceptions can be generated automatically. This is not yet possible and there is no literature of the automatic generation of misconceptions. The overlay model is a model of the part of the student knowledge which overlays the knowledge possessed by the system. With some modifications, this method seems to be the most appropriate for our purpose. There are, however, some simple and ad hoc methods introduced in the literature worthy of discussion.

A predicate, such as student-knows, can be used to construct a body of assertions about each of the concepts in the domain
that the student knows. For instance, if the system is teaching Prolog and the system by any means realises that the student knows how recursion works, this predicate is used to add the assertion:

student-knows(recursion)

into the student model. Taking the same simple approach, the student modelling component will view the things that the student knows incorrectly as what the student doesn't know, and take remedial action accordingly. A predicate student-doesn't-know can make assertions about the concepts that the student does not have a correct understanding of, or has no understanding at all. In such a case

student-doesn't-know(recursion)

is added to the previous assertions in the student model. If a conflict arises, either between two assertions, both inferred by the system or where one is a belief (i.e. volunteered by the user) and one inferred, then this inconsistency must be resolved.

Peachey and McCalla [1986] discuss a similar approach. They use a predicate for indicating that the student knows a concept, and the same predicate for indicating that the student knows an incorrect form of a concept. They give instances of planning rules constructed of conditions on occurrences of assertions and logical connectives (and, or, not) to control the presentation of concepts during a tutoring session. Figure 5.1 shows examples of such planning rules [Peachy and McCalla, 86].

Gable and Page [1980] discuss a more quantitative student model used in Generative CAI systems. A probabilistic grammar
using some rewrite rules which fire according to the proficiency level of the student is used for generating problems. An example of such a grammar and a sample derivation are shown in Figure 5.2. The proficiency level causes the problem generation to be altered for the individual student. For selecting a concept for presentation to the student the polynomial,

\[ P = a \cdot P_1 + b \cdot P_2 + c \cdot P_3 + d \cdot P_4 + e \cdot P_5 + f \cdot P_6 \]

is used. In this polynomial \( a-f \) are numbers between -1 and 1, and \( P_1-P_6 \) are parameters reflecting student's level, time a concept has been waiting to be presented, number of prerequisite concepts, and other related specifics of concepts and student. The concept selected is the one with largest \( P \) value. The student's proficiency
level, and changes in his level, control the remedial actions which are in the form of templates for explaining the problems. It should be noted that a context free probabilistic grammar is one in which rewrite rules have a single non-terminal on the left and each has probability associated and the generate.

Yet another way of modelling the student is to assume different levels of mastery for the student, in a range of -3 to +3, as suggested in Heines and O'Shea [1985]). In addition, two other

<table>
<thead>
<tr>
<th>Generation Probability</th>
<th>Proficiency level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>R0 : S</td>
<td>A</td>
</tr>
<tr>
<td>R1 : A</td>
<td>A* A</td>
</tr>
<tr>
<td>R2 : A</td>
<td>(-A)</td>
</tr>
<tr>
<td>R2 : A</td>
<td>p</td>
</tr>
<tr>
<td>R3 : A</td>
<td>q</td>
</tr>
<tr>
<td>R4 : A</td>
<td>r</td>
</tr>
<tr>
<td>R5 : *</td>
<td>&amp;</td>
</tr>
<tr>
<td>R6 : *</td>
<td>V</td>
</tr>
</tbody>
</table>

(1) S
(2) A
(3) A* A
(4) (-A)* A
(5) (-p)* (A* A)
(6) (-p)* (A* A)
(7) (-p)* (q r)
(8) (-p)* (q r)
(9) (-p)* (q r)

Figure 5.2 An Example of a Probabilistic Grammar and a Sample Derivation [Gable and Page, 76]

components, the learning rate and learning style, are present in this student model. The presentation of material will be faster or slower according to student's learning rate. And the learning style is used to choose the approach used in teaching. Recall that these
were two of the parameters identified as necessary for individualising instruction in our discussion of Section 5.2.

Self [1974] proposes the use of student models which take the form of several procedures. The answers produced by these procedures are supposed to be those given by the student. As the student learns, these procedures are also modified.

In this section some adhoc ways of modelling a student have been discussed. In the coming sections, two main approaches to student modelling, namely, overlay models and models based on student's misconceptions will be discussed.

5.4 The Overlay Model

5.4.1 General

The overlay model views the student model as an evolving description of what the student knows. The simplest way to illustrate such a view in this context is to have a semantic network of the knowledge in the domain and to gradually mark what the student knows. As noted by [Clancy 79], overlay models were proposed by [Carr and Goldstein, 77] and used in their tutor called WUMPUS. Other systems such as GUIDON [Clancy 87] and UMFE [Sleeman 85] have been based on this concept of overlays. We have adopted the overlay modelling approach in this study, and view the student knowledge as a subset of the systems knowledge.
The overlay model uses several sources of evidence to keep the overlay model up to date. These sources are [Wenger, 88]:

- Implicit evidence (behaviour of the Student)
- Explicit evidence (questioning or testing the student)
- Structural evidence (a syllabus of the material)
- Background evidence (initial contents of the model)

Limitations of overlay models pointed out by Wenger [1988] are:

1. The problem of representation of student knowledge as an overlay of system knowledge, which is a general problem not specific to overlay models.

2. The problem of no provisions for incorrect knowledge possessed by the student.

Because the student knowledge has to be a subset of the system knowledge, if the student solves a problem in a way not known by the system, a problem obviously arises. The second problem has given rise to the study of student misconceptions, and the creation of bug catalogues to incorporate into student models. The next section is devoted to the study of related methods. In the following sub-sections the overlay models used in GUIDON and UMFE will be studied. However, Overlay models have been used in other systems (e.g. ATR trainer [French, 90]).
5.4.2 GUIDON

Many aspects of the overlay model were employed by a system called WUSOR-II [Carr 77], which is a game tutor (the game is called WUMPUS) [Wenger 87]. Other ICAI systems have used this approach to student modelling, but we have chosen to study GUIDON which is an ICAI system based on the MYCIN medical expert system. We will then discuss UMFE which is a general purpose student modelling program based on the overlay modelling approach.

GUIDON considers a rule of MYCIN as a piece of knowledge, and that the student only knows it, or can use it, or there is evidence that he has applied it in solving a problem. In GUIDON the terms USE-HISTORY, SAPPLIED? and USED? are used to indicate each situation. The student state is evaluated with respect to MYCIN rules. The student knowledge is believed to be a subset of MYCIN knowledge, as depicted in Figure 5.4 [Clancy 87], even though it may be argued that this does not totally agree with reality. For maintaining and revising the student model, GUIDON uses background explicit or implicit evidence.

Background evidence which initialises the model could be information from previous sessions with the same student called USE-HISTORY. If no such information exists to initialise the model, the following criteria are used by GUIDON to decide about rules:
I. Comparing the student's experience to rule sophistication levels determined by the rule authors.

2. Deciding with high certainty that the student knows a definitional rule which has a sophistication level less than or equal to the student's level.

3. Assuming that the student doesn't know the rules which contain quantification clauses.

Evidence not taken into account by GUIDON which could better serve the purpose are analogies, generalisation-specialisation links, and prerequisite ties, as stated by Clancy [1987].
Implicit evidence is collected from the student’s behaviour or what he becomes interested in. This could be the data or help he requires. Asking for help could cause the system to reduce its belief that the student knows a domain rule.

Explicit evidence in GUIDON is collected by asking the student questions directly. Figure 5.5 shows the three components of the student model in GUIDON and how they are related to each other. These components are present and they are given a certainty factor for each rule in the MYCIN knowledge base. As the figure shows, the use history is the component persisting from session to session which gives an indication that the student knows certain rules. SAPPLIED on the other hand shows the rules that the student has applied. Finally, USED is the rules that the student has used to form a hypothesis, so, if asked to support his hypothesis, he will refer to this rule.

This subsection has described very briefly the approach taken by GUIDON using the overlay model. [Clancy 87] is a good reference for a detailed treatment of this approach. The next subsection will describe the UMFE approach to overlay modelling.
5.4.3 UMFE

UMFE [Sleeman 85] is a student modelling system based on the overlay modelling approach explained earlier. UMFE is not a component of any particular ICAI system, but it is claimed that it could be used as the student modelling component of ICAI systems. It infers using some rules, the topics that the student is believed to know. It also resolves conflicts when necessary, as described below.

As in GUIDON, in some situations UMFE asks the student whether or not she knows a topic and the student's answer is taken seriously. UMFE developers make some assumptions but claim that the approach taken in UMFE is applicable to domains where their assumptions do not hold.

UMFE is a data-driven user modelling system which can be used as a subsystem of an ICAI system. The topics in the domain of application have to be given to the system, along with their level of importance and difficulty. A set of inference rules indicating the hierarchical relationship among the topics are also given to the system.

Initially the system assumes no information about what the student knows and just asks her whether or not she knows the concept. The user is one source of information about herself while
the implicit and explicit inferences are two other sources available to UMFE. In implicit inference mode, UMFE assumes that concepts below the difficulty level that the student claims to know are known by the student. Explicit inference is information inferred by the system in the light of inference rules available to the system, an example of which was given in the previous paragraph. So, for each topic, the user model has an indication of whether the user knows, does not know, or there is no evidence either way. There is also an indication of the source of the evidence, which may be the user, implicit, or explicit, as discussed before.

Since there are multiple sources of evidence, conflicts are very likely to occur. UMFE resolves these by giving precedence to the user over explicit evidence, and in turn to explicit over implicit evidence. The system removes the part of inference rule which creates contradictions in the case of explicit inference and the system in a way learns. In the case of implicit inference, if the number of contradictions exceeds a threshold, this type of inference is stopped. If such a case takes place in explicit inference, or user contradictions exceed the corresponding threshold then the system, uses some so-called "witch hunt algorithms" to revise the model and remove inconsistencies.

5.4.4 The Knowledge Representation Issue and Genetic Graphs

5.4.4.1 Knowledge Representation
The information concerning the student is usually in the form of a database of quantities, sources of evidence and assertions about the student's state of knowledge. The Meta-knowledge of the model is usually in the form of production rules. Other formalisms, such as genetic graphs and procedural networks have also been used for representing the student model. An example is the use of such graphs and extensions in WUMPUS [Goldstein, 82].

5.4.4.2 Genetic Graphs

Genetic graphs were introduced by Goldstein [Wenger, 87; Goldstein, 82] and used in WUSOR. The word genetic should not be confused with its meaning in heredity. It originates in Piaget's idea of genetic epistemology which is the study of origins and development of a phenomenon. The nodes in the genetic graph represent domain knowledge and the student's knowledge is an overlay of the nodes. The links in the graph are evolutionary relationships of analogy, generalisation, specialisation, refinement and simplification. The student's learning progress is an overlay of these links. The graph may also be used for representing incorrect knowledge, by using deviation links.

Brecht and Jones [1988] have suggested some extensions to genetic graphs and applied them to domains of subtraction and ballet. Genetic graphs have not seen practical use other than that in WUSAR [Wenger, 87].

Figure 5.6 shows a part of the genetic graph used in WUSOR [Goldstein, 82]. Different types of links, namely generalisation and
specialisation, analogy, refinement and simplification are shown in the figure.

5.5 Models Based on Student's Misconceptions

It is possible to think of a student's knowledge as an overlay of domain knowledge possessed by the authoring shell. The student may also have some misconceptions not included in the correct knowledge of the shell. In other words the student may know something but incorrectly. It is not very easy to deal with such misconceptions. However, there is research aimed at foreseeing possible misconceptions of the student in different subject areas. [Burton and Brown, 82] discuss misconceptions in subtraction; [Stevens et al., 82] talk about types of conceptual bugs in student understandings of physical processes; Matz investigates student misconceptions in high school algebra. Others have taken limited domains such as Lisp [McCalla et al., 86; Farrell et al., 1984]
or PASCAL Programming [Johnson et. al., 83; Sopher et. al., 85]. The number of possible misconceptions, or bugs, is usually large, even for such a limited domain as simple arithmetic. Further, it would be very difficult if not impossible to automatically foresee all such misconceptions.
Two methods of student modelling based on student misconceptions seem to be the main approaches and will be considered here. They are perturbation modelling [Becker, 87], and differential models as used in WEST [Burton and Brown, 82]. The following subsections are devoted to these methods.

5.5.1. Perturbation Modelling

Perturbation modelling deals with student misconceptions by creating a perturbed model of the correct knowledge. This is done by having an incorrect version of the knowledge available to the system. This means that there will be an incorrect problem solver for each of the predictable misconceptions that a student may hold. The student answers are compared with those obtained from the incorrect solvers, and a match guides us to the misconception and the required remedial action. It should be noted that the possible misconceptions must be known beforehand.

Becker [1987] proposes some techniques for dealing with this method which will be discussed in what follows. He refers to perturbed tutoring module as Virtual Faulty Expert System (VFES). Student answers are checked against the answers generated by VFESs. This way student misconceptions are identified. Figure 5.7 graphically depicts this [Becker, 87]. In some cases the student's incorrect answer is the same as the correct answer produced by the system. In such cases, if the problem is decomposable, then the intermediate results are also taken into consideration in deciding about the misconception. Multiple misconceptions may also occur. These are, however, more difficult to detect. There may also be
misconceptions possessed by the student and unknown to the system, a problem identified by [Becker, 87].

[Becker, 87] suggests that problems must be generated or given to the system which can pinpoint misconceptions. In other words, the student modelling module should possess, or be able to generate, problems with answers capable of revealing student misconceptions. This was a brief discussion of perturbation modelling and for more details one should refer to [Becker, 87].

5.5.2 Differential Modelling

In Differential Modelling the student's action is compared with that of a computer based expert. Student misconceptions, or what the student has not mastered or does not know is isolated using the difference between student and expert behaviour. The difference may be skills that a student lacks to make the same decision as the expert, or pieces of knowledge that he lacks. An example would be the way a chess program makes its moves. A black box approach by itself would obviously not be appropriate for use in such a student modelling environment and a glass box expert or a combination of the two is necessary.

The differential modelling paradigm was proposed by [Burton and Brown, 82] and used in WEST. WEST is a coach used in a game environment. For playing the game several skills (arithmetic) each of
which is called an issue are required. Each issue can be thought of as a piece of the glass box expert used by the system. Two types of procedures called issue recognisors and issue evaluators are used with each skill or concept in WEST. The recognisors construct the model by comparing the student's move with that of the expert to identify the difference. Using the issue evaluators, the coach identifies the issue that the student lacks for making a better move (like the expert's). Remedial action can then take place, perhaps by providing some explanation to the student. Figure 5.8 [Burton and Brown, 82] illustrates the differential model construction on the basis of student behaviour. There are some problems with the differential modelling pointed out by Burton and
Figure 5.8 Construction of the Differential Model Used in WEST [Burton and Brown, 82]

Brown which are related to the selection of the issue requiring remediation, issues not independently recognised by the system, inconsistency of a student's behaviour and a student's use of alternative strategies. For details of these problems, one should refer to [Burton and Brown, 82].
5.5.3 Procedural Networks

A procedural network is a representation of decomposed procedural knowledge in the form of several sub procedures, starting from the general sub procedures at the top and finishing with primitive ones at the bottom, as seen in Figure 5.9. The links show the calling path among the nodes. A procedural network can be executed [Wenger, 87]. Each of the sub procedures in a procedural network may be replaced by a faulty version embodying a corresponding misconception. As many sub procedures as necessary are replaced to try to generate the same answer as the student's. The faulty sub procedures which are added to generate the same answer as the student's answer correspond to the misconceptions the student possesses.

5.6 Student Modelling for ITS Using the CR Model Approach

The reason for designing such a component is the need for generality in student modelling for authoring shells, which as we discussed before, are proposed as tools for developing intelligent tutoring systems, and this study is aimed at taking a step toward this ultimate goal. Like any other student modelling component, the system must be able to construct a model of the learner as she interacts with the system. The student modelling component needs to construct an initial model to base the sequencing of the material on. UMFE, as an example discussed earlier, makes no initial assumptions about a student's state of knowledge. The intelligent assistant constructed in this study as a component of the authoring
shell, can provide this information using the CR network and the retrieved related documents as discussed in the following section.

Figure 5.9 An Example of a Procedural Network for Subtraction [Good, 87]

5.6.1 Assessing the Student Based on the Concept Relationship Network

As indicated before, an authoring system must be given many of the specifics of the course to be taught such as text and questions. In the light of the information acquired during the knowledge acquisition sessions (see Chapter 3) and embodied in the CR network, the author has to do very little to construct the initial student models using the intelligent assistant. This is an advantage
of student modelling through the CR model. For each concept, domain knowledge is kept. For the case of geography, this knowledge will be mostly factual. In other cases, the CR model may have to be modified by having a small knowledge base, called a knowledge source, associated with it. Before we can construct an initial model of the student based on the concept relationship model approach, two assumptions will have to be made.

Assumption 1: If a student can identify the concepts which fall under a topic, and can further establish the relationships among these concepts correctly, then the student is very likely to know the topic. Without any further effort, if any of the identified concepts and the established relationships also belong to another topic, then that topic can also be considered partially known by the student.

Assumption 2: It is also implied by the first assumption that if a student cannot identify the concepts related to a topic and he cannot establish the relationships among these concepts, then we can conclude that he does not know the topic. If any of these concepts fall under another topic, then that topic is also not completely known by the student.

This is only one of the methods we have used for evaluating the student, and evidence obtained in this way is given a certainty of 20%. Thus, a student who identifies a concept and its relationships, but fails the other two tests, is not given credit for knowing that concept. A student who successfully completes the other tests, but
fails to identify the concept, possibly through unfamiliarity with the vocabulary employed, may still be identified by our system as "knowing" the concept. The two assumptions made here address the extreme cases. Cases between these two extremes are likely to be of more interest:

. The student may be able to identify the concepts but not be able to establish the relationships among them.

. The student may be able to identify some of the concepts and be able to establish some of the relationships correctly.

. The student may identify some concepts or establish some of the relationships among the concepts but have errors in doing so.

Each of the above cases can be identified by the system as it interacts with the student through its student interface (educational questioning has been considered for this purpose). The initial student model is constructed taking into consideration each of the concepts forming a customised CR model. This is done for every student interacting with the system. A quantitative measure of belief is recorded for each concept corresponding to the following range of measures of belief in student knowledge:

. Known
. Partially known
. Not known.
The target authoring system (or Authoring Shell) is expected to use this information to sequence the tutorial material and to select a suitable teaching strategy if it has provisions for multiple teaching strategies. The student model is, however, mainly used for deciding which concepts are to be covered and which not. So, each concept is treated differently according to the individual student's state of knowledge, which makes the learning environment an individualised learning one. It is not necessary to point out that these values can be changed during the course of tutoring by the target shell, if contradictory evidence arises.

5.6.2 Construction of the Initial Student Model

An ICAI system must rely on some kind of an initial student model. One way of initialising the model for each concept would be to present a question of average difficulty to the student and decide whether or not she knows that concept. Identifying such a question would not be easy to do and in many cases, and it may not be possible to make such a decision. So, in most cases the author has to provide such a question. The questioning should start at a concept of average difficulty, and an easier or more difficult concept selected from the outcome.

The retrieval component of the intelligent assistant retrieves documents related to each of the concepts. This is a good source to be used for testing a student's knowledge of the domain. As mentioned earlier, the CR network is used for testing the student's knowledge. Nodes of the CR network are marked as the student identifies and shows that he knows them. The retrieved text can be
Governments that were inclined towards socialism were more directly involved in their national economies than were those of the more capitalist philosophy.

Figure 5.10 A Question Generated From Text by IACP

used to confirm the student's knowledge of the concepts. This process starts at a concept with an average level of difficulty. The text is retrieved using a set of keywords defining each concept. Sentences in the text containing these keywords in the text are presented to the student with the keywords left as blanks to be filled in by the student. A list of synonyms is kept for each keyword and consulted if there is no match on the keyword in question. An example of such a question generated from retrieved text by IACP is shown in Figure 5.10. This method has been used in teaching of English [Lunzer et al., 84] and the sentences presented are known as clozes. However, there is a problem which is inherited from the retrieval of text. Some of the documents retrieved are going to be unrelated to the concept being considered. Thus the questions generated from such text are also going to be irrelevant. There are two ways of resolving this problem. One is to have the author go through the generated questions and delete the irrelevant ones. We have however chosen a second solution which is to let the student differentiate between relevant and irrelevant questions. A
percentage of the score is devoted to the student's understanding of what is relevant and what is not. The student has the option of entering NR (not related) in reply to such questions. This should not pose any problems as differentiating between relevant and irrelevant questions is also an indication of a student's level of understanding of a concept. The student modelling component should be checked by the teacher before use by the students. If the student modelling component is checked, irrelevant material can be deleted by the teacher. In this case, the possibility of the student entering NR, and having the system accept this answer, can be eliminated.

During the knowledge acquisition phase, the expert is asked to provide some key knowledge for each concept, as we explained in Chapter 2. This key knowledge is then formalised and used as a basis for the questions from the knowledge base. After the student attempts the generated questions, questions from the knowledge base are presented. No problem solving knowledge was acquired in the experiment with the intelligent assistant because the topic selected for experimentation was development in the context of geography. The knowledge is mostly textual in the domain of geography and specially the topic of development.

The knowledge was formalised and asserted into the knowledge base of the system which is in a Prolog-like language called the Logic Manager [LM,91]. Although the knowledge formalised in this case was textual, problem solving knowledge can also be formalised and used for the same purpose using the Logic Manager.
Once the student has attempted the two types of tests for a concept, he is asked to identify the concept, by presenting a list of concepts to the student and asking him to select from the list (Figure 5.11). Every time a new concept is identified by the student, he is asked to identify the relationship among the current and previously identified concepts (Figure 5.12). This is also done by presenting a list of possible relationships and asking him to select from the list. A score is given to each part of this testing process. The author can change the weights given to each part of the test. A reinforcement card is displayed each time the learner scores above a threshold for a concept.

![Select a concept and click on Ok](image)

Figure 5.11 A List of Concepts Displayed for the Student to Select From
You have identified Concepts government policies and distribution of wealth. Can you tell me what their relationship is?

[No] [Yes]

Figure 5.12 Students Asked to Identify Relationship Between Concepts

The student modelling component of the system is implemented using a Hypercard stack. Several Hypercard XCMDs (External Commands) written in C were used to do some necessary functions. The stack is updated automatically every time material for a course is generated by the intelligent assistant. The student interacts with this component, and models of students are generated and stored. This process is meant to take place while the author is selecting the material to be used in the course. The student modelling component allows the author, who is meant to be the teacher, to change the parameters related to scoring of the student through a menu called "Test".

Each concept has a set of prerequisites associated with it, as shown in Figure 5.13. This graph can be converted to a lesson hierarchy as shown in Figure 5.14. Concepts at the same level of hierarchy may be considered in any sequence. It is assumed that the prerequisites of a known concept are also known by the student. Concepts which are prerequisites of known concepts are marked as...
"inferred known". Concepts at higher levels of the hierarchy are also used for checking the student unless they are already marked "not known". The process of marking the nodes continues until all the nodes are marked. Hierarchy levels are acquired from the expert when the CR network is constructed.

A level of difficulty is associated with each concept in the domain. This is provided by an expert in the domain, who could well be a teacher. The process of assigning tags to concepts starts at a concept with an average level of difficulty and continues from there.
Figure 5.13 Post and Prerequisite Concepts Represented as a Graph

Figure 5.14 A Lesson Hierarchy Obtained from the Pre/Postrequisite Graph
Concepts are tagged first through the CR network and the assigned tags are confirmed or a conflict may arise. If there is a contradiction between the results obtained using the "fill in the blanks" tests and the CR network, it is resolved by tagging that node as "inferred partially known". Nodes which are already marked as partially known are left as they are.

A set of rules can be generated using the hierarchy and difficulty levels. These rules, as in UMFE, can infer concepts to be known by the student, based on what the student knows. That is, assuming that the levels of hierarchy and difficulty of concepts C1, C2 and C3 is lower than that of concept C4, then if the student knows concepts C1, C2, C3, we infer that she knows C4. The reverse is also true for concepts not known by the student. A tag of "inferred known" or "inferred not known" may be given to the corresponding concepts depending on the knowledge of the student known by the system.

In student modelling, a certainty factor is sometimes associated with inferred information. If there is a conflict among inferred information about a concept then the one with higher certainty is taken to be closer to truth. Explicit information is considered to be more reliable than inferred information.

Other aspects of student performance are: the time the student spends before answering questions or solving problems, and the number of questions and amount of data she requests. A threshold value can be assumed for time allowed for questions and
problems, which is a function of the difficulty level and length of the problem or question. The amount of data requested is an indication of whether or not the student needs help. A set of actions is associated with each of the conditions related to these aspects of a student's performance. These can be in the form of rules related to each concept. The conditions would be limits on the thresholds and the actions would be what is to be done. This type of assessment was not included in IACP.

The model constructed through the CR network is meant to be an initial model. The target system will have to update, maintain, and perhaps correct this initial model. The model is simple, but requires little effort on the part of the author. Another advantage of student modelling using CR networks is that, being concept based, they are reusable.
6. An Intelligent Assistant for Courseware Preparation

6.1 General

In the previous chapters we discussed the theoretical basis for building an intelligent assistant for courseware preparation. This chapter presents IACP, the intelligent assistant for courseware preparation. In Chapter 3, we explained the CR model developed and used in this study for representing the structure of a domain. Based on this model we have constructed an intelligent assistant (IACP) which assists in the preparation of material needed for developing a course. The output of IACP can be used in both conventional CAI and ITS environments where a student model is necessary. It can even be used on its own as a tool for learning if exploratory learning or semiguided learning is the chosen environment for instruction. Environments are a common topic in ITS research. In a learning environment, a student explores the environment and researchers like [Papert, 80] suggest that this is all that is needed for learning. In ITSs, the system guides the learner through the material in an intelligent manner, based on a model it constructs of the learner.

6.1.1 An overview of IACP

IACP is an intelligent assistant for courseware preparation. Although the field of ITS is getting a great deal of attention from the computer science, psychology, and linguistics research communities, there are no tools available which would enable a non-programmer (typically a teacher) to construct an ITS. This
may partly be due to the fact that the field is still experimental. The cost of building an ITS is high and it is practically very difficult, if not impossible, for a nonprogrammer to build one. As a step toward a general tool for building an ITS, IACP automates some of the important tasks. It achieves its goal using a model of the domain structure. IACP is a tool meant for non-programmers. A knowledge engineer must, however, be involved in the development of a model of the structure of the domain of interest. This need only be done once, and possibly by someone other than the author of the courseware being created. IACP through its various operational components provides an environment for teachers to rapidly put together material for creating courseware. To make the material easily inspectable by the teacher and usable by the students, hypertext techniques are developed and used. A student model is a central component of an ITS. IACP generates a student modelling component which can be used for generating models of the students interacting with the system.

6.2 Courseware Design

Instructional design usually involves organising large amounts of information. In doing this time-consuming task, the instructional designer must,

1. Analyse the domain of knowledge and decide what to include in the course,
2. Organise the material according to instructional goals.
3. Produce the final form of the course (lessons, tests, etc.)
It is probably impossible to analyse all the material available for a domain and reach a decision for step 1 of this process. One may analyse a portion of this material. Even doing this would be an extremely tedious and time consuming task. The general structure of domain knowledge would make the first step above easier for the instructional designer. By structure, we mean the concepts forming the domain, how they are related, their place in the structure and how they can be combined to form a course. In Chapter 3 we introduced a structure representation formalism and a methodology for eliciting the structure of a domain. The CR model introduced in Chapter 3 depicts the structure of a domain and is constructed through interviews with an expert familiar with subject matter. The structure of the domain represented as a CR network can be used for analysing the domain. It can be used as a basis for selecting what to be included in a course.

As mentioned earlier, the teacher using the system has the opportunity, and this is recommended, to go through the generated hypertext and eliminate the unrelated ones, or even to add relevant notes and comments. Even if all the material is relevant, the teacher may decide to exclude some. The multi-layered hypertext generated by IACP provides an easy-to-follow navigation tool for the selection process.

The material that is to be used for teaching has to be organised according to the instructional model used. In CAI systems, material is often structured in a predetermined sequence and the student is guided through this sequence. Branches are sometimes made during the instruction depending on a student's
answers. In generative CAI, problems are generated by the system. In intelligent tutoring systems, the material is sequenced with respect to a model of the learner, which is maintained by the system.

Hypertext systems were not initially intended for education. Hypertext has, however, attracted attention for use as an educational tool [Jonassen 91; Jonassen 90; Raper, Green 89; Intermedia]. Electronic books based on hypertext techniques are also useful educational tools. The material structured by IACP is linked together using its automatic hypertext linking facility. The material is linked in such a way that the designer of courseware can easily navigate through and select/validate the material she desires to use in the course. This hypertext environment (enhancable to a hypermedia) can be used for instruction as it stands. We have introduced the notion of intelligent links. By adding the intelligence to the links, the environment is turned into a semi-guided learning environment, as explained later in this chapter. Thus, IACP is both an aid in performing step 2 or fully automates step 2, depending on the desired environment.

An authoring system helps a course designer put a conventional CAI course in its final form. This is what we expect of the authoring shells for producing ITSs in addition to providing assistance in the first two steps in the design of courseware. IACP assumes the final step to be the job of the target authoring tool used for putting the course in its final form.
In summary, IACP fully supports the first step in the development of courseware. The second step may or may not be fully supported by IACP. This depends on the design objectives. The extent of support for step 3 is also dependant on the type of the environment the designer is aiming to achieve.

6.3 IACP as a tool for Courseware Preparation

A model of the domain structure constitutes the backbone of IACP. Before the end users (teachers) can use the system, the backbone of IACP must be present. We have developed the CR model for representing the structure of the domain. We explained a methodology for formalising the domain structure knowledge through interviews with an expert.

Three types of users may interact with IACP. This was depicted in Figure 1.4. We have classified the users of IACP as follows:

Super user
Author
Student.

A super user is a knowledge engineer who together with an expert develops a CR model, formalises and enters it into IACP, and maintains the CR model. An author is a user who utilises IACP for courseware preparation. If structure elicitation were to be automated, an expert could play the role of a super user, depending on the extent and effectiveness of the automation. The author uses
IACP for the final goal of developing a course. Depending on the
design in the mind of the author, the role of the learners as users
of IACP changes. They may use the system for the sole purpose of
student model construction for use in the final ITS being built.
They may be allowed to explore the material prepared by the
system in order to learn. Hypertext is a tool for learning as
discussed in Chapter 4.

6.4 Operation of IACP

6.4.1 Starting IACP

Figure 6.1 shows the various code modules of IACP. Hypercard
enhanced with a number of programs written in C, called external
commands, must be available. Note that the necessary external
commands must be added to Hypercard before running IACP.
External commands were imported into Hypercard using a resource
mover. There are three Hypercard stacks shown in the figure. The
code of various components of IACP reside in these stacks. The
stacks also contain various templates, one of which is a student
modelling component template. This template contains a general
student modelling component’s code, customised for the course
being developed. This template is called a student modelling
template. Another template is the template of a stack used for
organising the resources of a concept. This is called the concept
resource template. Various other templates of individual links,
displays and information fields are also used. We will not go into
the details of these. The code of Logic Manager is kept in a file
called LMCode. This file must be present in the same folder as the
generated student modelling component when this module is in use.
The student modeller uses the Logic Manager as its inference
engine. The three stacks shown in Figure 6.1 are: the CR network,
Tbg, SMKB.

Figure 6.1 Files and Stacks Needed for Starting IACP

To start IACP, the three stacks and the LMCode file must be
present. The user double-clicks on CR network icon and a welcome
screen appears (Figure 6.2). IACP's interface is a mouse and menu
Welcome to the Hypertext Based Intelligent Assistant for Courseware Preparation (HBIA)

click to start

Figure 6.2 IACP's Welcome Screen

interface. Clicking the mouse on this screen takes the user into the CR network earlier input into the system. The menu bar of IACP has an Assistance menu. This menu provides the user with the following options:

Customise
Retrieve
Open Database
Close Database
Append Database
Reset
Help
The Customise option customises the CR network to the author's needs. Retrieve is used to retrieve and link resources once the CR network has been customised. Open Database is used to open the textbase. Close Database closes an open textbase and Append Database appends to it. Reset is used to reset the system after the operation of the system is interrupted in the middle of a run. Help provides explanations for each of these menu items. In the following section we will explain how IACP is used for preparation of courseware.

6.4.2 IACP in Operation

In using IACP different situations may arise. In the best case, the structure of the domain in the form of a CR model will be available. It may be prepared by the education department, or built by a knowledge engineer participating in a courseware development project. The CR model is constructed once and maintained and used over time for preparing course material. Maintenance operations are performed on the basis of suggestions from the users and information provided by the system itself. In summary, two cases may arise in the operation of IACP with respect to the availability of a mode of the domain:

1. A CR model of the domain exists
2. A CR model of the domain has to be constructed

The following set of steps must be followed for using IACP for developing courseware:

1. a. Development of a CR model of the domain
b. Customisation of the CR model with respect to the instructional goals

2. Execution
3. Selection
4. Importation

Step 1.a is skipped if a CR model already exists (situation 1). These steps are followed linearly and the author goes from one step to the next.

The author first has to see if a CR model of the domain exists. If the idea of courseware construction based on CR models finds wide acceptance in educational circles, models in various fields will be readily available for teacher use. Education departments are among the potential providers of such services for their teachers. Private organisations providing courseware material support over a computer network can best take advantage of CR models. In this case, such models will be privately developed. At present, to use IACP, one must first construct a CR model of the domain of interest. The task of structure elicitation has not been automated in IACP. This means that a knowledge engineer would have to construct the model through manual interviews with the expert. This model is the backbone of the system, making the task of constructing it a critical one requiring time and commitment. The CR model is not constructed with the instructional goals of a particular author in mind. Each author can, however, customise the model according to personal needs.
Customisation is the first step in the operation of IACP in the presence of a previously constructed CR model, which will reflect the mental model of the expert being interviewed. This is why we have provided the customisation facility, to alter it for the author. The learners' mental models are also acquired by the student modelling module generated by IACP. Student mental models are transformed as much as possible into models similar to that of the expert through instruction.

IACP provides a menu option for customisation of its CR model. Selection of this option will cause the changes made to the model to take effect. The customised version of the model is then created for the author to use through the succeeding steps in the operation of IACP. A variety of changes can be made to a CR network. The changes can range from changing a descriptor item, or the entire descriptor of a concept, to the addition or deletion of concepts. To select such options, the user holds the mouse button down within the concept node of interest. A menu containing the following set of options then pops up:

List All Concepts
Display Descriptor
Create Descriptor
Edit Descriptor
Select Concept
AutoSelect
Select All Concepts
Add Concept
Delete Concept
Input Knowledge
Append to Knowledge Base
Text Stack

The first option causes a list of concepts to be displayed. The author will need to view this list at times. The list disappears with a mouse click. Menu option "Edit Descriptor" allows an author to make changes to one or all items of a descriptor. Before making the changes, the author may wish to view the contents of the descriptor. This may be done by selecting menu option "Display Descriptor". An author may want to add or delete a concept. Deletion is simple. The author selects "Delete Concept" while the mouse is within the concept to be deleted. Adding a concept is done in a similar way by holding the mouse button down and selecting menu option "Add Concept". Questions regarding the name of the concept being added and its relationship with the other concepts must then be answered. A call to "Create Descriptor" creates a descriptor for the corresponding node. There are two options related to input and further appending key knowledge related to a concept. These are "Input Knowledge" and "Append To Knowledge Base" menu items. Other important menu items are the concept selection items. The other options discussed above are used after the application of "Customise" menu option. Selection menu options are used prior to "Customise". The entire network can be selected using the "Select All Concepts" menu item. This will cause all the concepts to be selected at once. Individual concepts can be selected using "Select Concept menu item. "AutoSelect" allows the author to select an concept and the degree of relationship and let the system automatically select all related concepts. Once the
process of selection is complete, the author can finalise the
customisation, by selecting the menu option "Customise" from the
"Assistance " menu on the menubar. An author will then be taken to
the customised copy of the network where she can make further
modifications.

The second step in the operation of IACP is the execution
step. The model is executed to prepare material necessary for
courseware construction. This operation is started from the
customised model. There are certain conditions that must be met
before this step. An indexed file containing the documents stored
for this purpose must be open and available to IACP. If no such file
exists it must be created. The "Assistance" menu provides options
for this purpose. Several files may be appended into one file using
"Append to Database" option of this menu. "Build index" can be used
to index the resulting file. Two files are generated as a result of
the indexing operation. One is a key file and the other is a pointer
file designated by extensions "k" and "p". "Open Database" and
"Close Database" open and close the document file and the index
files. Once the database is open the system can start retrieving
using the customised CR model constructed in the previous step.
This is done by selecting "Retrieve" from the "Assistance" menu.
This step involves the retrieval, linking and structuring of the
retrieved material. Material is retrieved and converted into
hypertext. The operation of hypertext generation was explained in
detail in Chapter 4. We will leave more details of the retrieval
component to the following sections. This step takes time and the
author may have to let this operation go on for several hours. For a
simple customised network with approximately 10 concepts it takes about 3 hours to complete this step on a Macintosh Classic II.

The third step in the operation of IACP is selection. The selection step in the operation of IACP corresponds to the second step in the development of courseware discussed earlier in this chapter. The material to be included in a course has been retrieved and organised into a multi-layered hypertext. The material retrieved may not all be relevant. Even if it is all relevant, the author may not want to include it all in her course. The hypertext provides the teacher with an easy to follow navigation tool for going through the material and making decisions about the inclusion and exclusion of material. The author may want to add her own comments at various points. All these operations are supported and are available to the author. In addition to the hypertext generated by IACP, a student modeller is also generated. This component may be checked by the author for verification of the generated questions. This should be done because some of the questions are generated from retrieved text, and may be irrelevant due to retrieval precision, or improper because the author feels so. It is, however, possible to use this module without making any changes. If so, the students are then allowed to enter, "not relevant" where they feel the question is not relevant. The system accounts for and allows such answers. It is recommended that the author verify the student modelling component before putting it into the hands of the students. In any case, while the author is going through the hypertext (or hypermedia if that is the case) and selecting and preparing the material, the student modelling component generates models of the students using it. This process
is only carried out when the material is prepared for use in an ITS environment or the hypertext is the final product and intelligent document level links are necessary.

Importation is the preliminary step for putting a course in its final form, which is the third step in the courseware development scenario discussed earlier in this chapter. In this step the material verified and modified by the author is ported to the final courseware development environment. If the tool used is Hypercard, there is no need for any extra effort. Otherwise, transformations in the format of the material may be necessary. If the goal of the author is simply to create a self guided discovery environment in the form of a hypertext for teaching, this final step is not necessary.

The result of following the above four steps is a personal folder with the author's name containing a customised CR model, a set of retrieved, linked and verified resources for each of the concepts in the CR network. These are in the form of hypertext linked at a higher level through the CR model, and a verified student modelling module for generating models of learners.

6.5 The Architecture of IACP

Both a simplified and a more detailed view of the architecture of IACP were presented in Chapter 1. As shown in Figure 1.9, the following components are the building blocks of IACP:
A conceptual model
A knowledge base
Thesauruses
An inference engine
A user interface
A hyperbase
A student model generator
A retrieval component

Some of these components have already been explained. We will only briefly review them here.

6.5.1 The Conceptual Model

A conceptual model of the domain is the backbone of IACP. The CR models introduced in Chapter 3 were developed for this purpose. They depict the structure of the domain, and are used as a medium for retrieval, structuring and automatic construction of hypertexts. Student models are also constructed through the CR model as explained in earlier chapters. A CR model must be present before IACP can operate.

6.5.2 The Knowledge Base and the Inference Engine

The knowledge base is the collection of key knowledge chunks related to each of the concepts. It can also be used for future expansion and inclusion of other knowledge for improving various components of the system. The knowledge base is filled when the system starts operation. The knowledge contained in the knowledge
base is domain specific knowledge. It is used by the inference engine to generate questions and their answers. The questions generated through the knowledge base are used along with other questions generated from text and the domain model to generate a model of the student.

The inference engine is used to manipulate the knowledge contained in the knowledge base in order to generate questions and answers to the questions. Logic Manager [Logic Manager 91a], a version of prolog which is used from within Hypercard was used to construct the inference engine of IACP. It operates through a Hypercard external command. Logic Manager is a product of an Apple research group called the Advanced Technology Group. We will explain the Logic Manager later in this chapter.

6.5.3 Thesauruses

These are domain specific thesauruses used at various points in the operation of IACP. Synonyms for concepts are one type of items found in the thesauruses. Another type of items are the keyword synonyms. Synonyms are used in the verification of the CR model. They help find redundancies in the model. Another use of the synonyms of is in modelling the student. Synonyms of keywords are useful in retrieval of related material by the retrieval component. They are also used in the student modelling component for checking the student's answers. The automatic hypertext generator uses them for generating more links.
6.5.4 The User Interface

The user interface is responsible for presentation of system messages to the user. It takes the user's responses and translates them into a form understandable by IACP. The user interface is implemented in Hypertalk, the language of Hypercard. Where necessary, external commands written in C were used. The user interface has been designed to interact with all three types of users mentioned earlier. The interface of IACP is a friendly mouse and menu interface. In most cases the user makes his choice from menus or lists using the mouse. Examples of such interactions through the user interface were presented in the previous sections.

6.5.5 The Hypertext Component

The hyperdocument collection is a set of interlinked document collections containing the retrieved documents. The hyperdocument clusters are created by the hypertext generator component of IACP using the CR model. Each cluster is a cohesive set of documents, put in the same cluster because they are all related to the same concept. The material contained in each of the document collections is interlinked. The basis of this linking is the information contained in the CR model. We refer to the links at this level as internal links. Each cluster is connected to all other clusters which have documents related to its documents. These links are referred to as external links. In short, an internal link connects related documents in the same cluster and an external link connects related documents in different clusters. We refer to
both internal and external links as document level links. As we have explained before, there is another level of linking, at the concept relationship network level. Each cluster in the hyperdocument base is connected to its corresponding concept in the CR network. This multi-level linking was explained in earlier chapters. The material in the clusters can potentially be of different media. Depending on the type of media the hyperbase will be a hypertext or a hypermedia.

6.5.6 The Student Modelling Component

This component generates the module used for modelling the learners. It does this using the CR network, the knowledge base, and the templates contained in IACP. The code and information contained in the template is adapted according to the CR model. The student modelling component and its generation and use were discussed in Chapter 5.

6.5.7 The Retrieval Component

The retrieval component is an important component of IACP, but we have deferred a detailed explanation of it to this chapter. In what follows, text retrieval, as it relates to this work, will be discussed. An explanation of the retrieval component of IACP will follow that.
6.5.7.1 Text Retrieval

Retrieval effectiveness affects the performance of IACP. The verification done in the operation of IACP is to overcome the problems of current retrieval methods. In this step the author goes through the retrieved and structured material and eliminates unrelated and unwanted material. Even if all the material retrieved were relevant, the verification step would still be necessary. This is because the author may not want something included in her course even if it is relevant. The more effective the retrieval, the less time is spent in this step. The verification step is supported by a hypertext environment automatically created by IACP.

Most commercially available text retrieval systems are based on an index of words or phrases defining the contents of the documents. To retrieve documents, Boolean logic based queries can be constructed using Boolean operators AND OR and NOT to link words and form queries. Documents relevant to a query are retrieved.

Two approaches are common in text retrieval systems. One is a keyword based approach and the other is a full text based approach. In keyword based systems the indexing of documents is usually done manually. Most library systems are of this type. Human experts assign keywords to each of document and retrieval is based on matching of queries with these keywords. A new document entering the system must be indexed in the same way. In IACP, keywords are assigned to concepts rather than documents. This makes addition of documents much easier in IACP. In full text
retrieval, every word in a document is examined. Unlike the previous method, index building in full text retrieval systems is done automatically. Queries in full text retrieval systems are also built using Boolean operators.

Keyword based systems require manual assignment of keywords to documents. This makes the addition of new documents expensive. The problem with full text retrieval systems is that the construction of an index of all words in a document takes up a lot of space. The size of the index is usually as large as the document itself. This is in addition to the general problem of forming the queries in such a way as to get a balance between retrieving documents that are both relevant and as large a proportion of relevant documents as possible. The problem of space is not of much concern, as storage devices such as CD ROMs can store large amounts of information.

To improve retrieval effectiveness, various methods have been suggested. One improvement has been through the assignment of weights to the terms in a query. Documents satisfying a threshold can then be retrieved. In IACP it is possible to assign weights and set a retrieval threshold. The weights are assigned to keywords of each concept by the expert assigning the keywords to concepts. The higher the weight assigned to a keyword the more the significance of that keyword. It has been suggested that the frequency of occurrence of a term is an indication of its significance [Robertson, Sprack-Jones 76]. User feedback has also been used as a mechanism for improving retrieval effectiveness [Wu, Salton, 81]. Frequency of occurrence of the words in a
document is believed to carry information on the relevance of a document to a query. Approximate string matching and clustering techniques have also been proposed to achieve better retrieval. These methods are classified under the statistical approach to information retrieval. In this approach, text is represented based on counts of words and retrieval is based on similarity functions. Systems built based on this approach are independent of the domain. In IACP the expert assigns weights to keywords with the specific goal of courseware preparation in mind. Logical operators have also been assigned weights as another variation on Boolean logic based searching [Salton 84]. Extensions such as fuzzy logic have also been proposed as possible improvements.

The text understanding research community would probably believe that for any retrieval to be acceptable, it must be based on a full understanding of the document. There are, however, no such systems commercially available. There has been some success in limited domains such as understanding stock exchange messages or analysing road traffic accidents [Jones, 90]. Full machine understanding of text is costly and requires a lot of domain specific knowledge. Understanding of general text without specifying the domain is probably impossible. As mentioned earlier, in keyword based retrieval systems, a human expert reads text and associates keywords with it. Although the applicability of CR models as an improvement to the retrieval methods is beyond the scope of this study, it is a model that could be tried as a middle road between retrieval based on keywords and retrieval based on full understanding of text.
An approach reported in the literature based on a partial understanding of text is called text skimming [DeJong 79]. This approach was the basis for a system called FERRET [Mauldin, 91], which uses a partial understanding of text aimed at more effective retrieval. FERRET scans text and converts it into an abstract. Queries are then matched against these abstracts. FERRET adapts its text skimming parser with respect to feedback received from the user on precision of the retrieved documents. AIDA [Jones, 90] an Australian project of the Computer Power Group based in Canberra is an earlier system than FERRET. AIDA uses knowledge based text processing techniques with a great degree of success to generate abstracts of documents which can then be used for retrieval.

Various AI techniques have been applied for enhancing the performance of information retrieval systems [Chen, Dhar, 91; Croft, Thompson, 87; Croft, 93]. Several knowledge based systems taking advantage of expert systems technology have been constructed [Monarch, Carbonell, 87; Cohen, Kjeldson, 87; Shoval, 85; Fox, 87; Croft, Thompson, 87; Biswas et. al., 85; McCune et. al. 85]. A combination of statistical and AI approaches have also been tried [Croft, 93]. Knowledge based retrieval systems are currently expensive to implement and are only effective in narrow and specific domains [Croft, 93].

The effectiveness of retrieval affects the performance of IACP. Better recall enriches the material that can be put together for a course. Better precision reduces the amount of work the author has to do in the selection stage. The architecture of IACP is
flexible and there is room for the inclusion of better retrieval functions once they are available and applicable to various domains.

6.5.7.2 Retrieval of Documents in IACP

The retrieval component uses an index of all words in the document base. The indexing is done automatically prior to retrieval. Several menu options are provided for this purpose as explained previously. Frequency of occurrence of words is also calculated. The user does not have to assist in the retrieval process. Documents are retrieved based on the knowledge contained in the CR model. Messages are, however, displayed during the long process of retrieval and linking and organisation and the user is informed of progress. The user only selects the concepts of interest to him and does not have to worry about the details of the retrieval process, but simply selects the "Retrieve" menu option from the "Assistance" menu. Once the retrieval is complete, the selection step in the operation of the system starts. This is necessary because a 100% precision with reasonable recall is impossible with the current technology. In this step, the user, aided by hypertext techniques, selects from the retrieved documents those to be included in the course being constructed.

6.5.7.2.1 Recall and Precision

The documents retrieved are usually not all related and comprise only a portion of the relevant documents. Precision is the ratio of relevant documents retrieved to the total number of
documents retrieved. Recall is the ratio of the number of relevant documents retrieved to the total number of relevant documents in the textbase. Depending on the application, a user may care more for precision or recall. Recall is of more importance in IACP. Recall can be increased by the use of synonyms [Chen et al, 93]. IACP achieves its goal through the use of many carefully formulated queries including a large number of keywords from the CR model, and the use of synonyms. We cannot afford to have any unrelated documents in the collection used for constructing a course. A recall of 100% is almost impossible with the level of generality IACP is aiming for. Thus, the selection step had to be included in the operation of IACP to eliminate unrelated or unwanted documents. This process has been made easier through the use of hypertext techniques mentioned before.

The user of IACP does not have to worry about the details of retrieval. Queries do not have to be formulated as this is made a responsibility of the retrieval component. All selected concepts are automatically included in the retrieval.

6.5.7.2.2 Proximity Searching

This capability of the retrieval component allows words to be searched for within a certain neighbourhood of one another. The distance between the words can be selected by the user. This is done through the "Neighbourhood Size" option of the "Text" menu.
6.5.7.2.3 Similarity of Documents

Document clustering is a known method in information retrieval. Documents are clustered around a centroid with respect to their similarity. This similarity is calculated using a similarity function. For a discussion of this topic [Salton 75] is a good source. In the retrieval component of IACP similarity is used as a measure of relationship between documents for a slightly different purpose. Similarity between documents of different clusters is used for improving the CR model. The strength of relationships between concepts can be evaluated and the model can be modified on the basis of the similarity of their document clusters.

Similarity between two document sets is computed by forming a vector containing words in common for each set of documents related to each of the concepts. These vectors are then compared. The number of words in common divided by the total number of words in the smaller of the two vectors is the similarity value used as a basis for the learning function.

Concepts in the CR model are defined partially by keywords, each of which has a weight associated with it. Concept nodes are connected through relationship nodes. Each concept is given a relationship level by the expert when the CR model is constructed. This initial level is obviously not quantitative. The level of relationship between two concepts corresponds to a value between 0 and 1. These levels of relationships and their corresponding numerical ranges are shown in Table 6.1. This measure is later used by the person maintaining the retrieval component to take the
required action. It is his decision to make changes, or not, to relationship strengths initially assigned by the expert.

Table 6.1 Levels and Ranges of Relationships

<table>
<thead>
<tr>
<th>Level of Relationship</th>
<th>Corresponding Numerical Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Weak</td>
<td>≥0.0 &lt;0.2</td>
</tr>
<tr>
<td>Weak</td>
<td>&gt;0.2 &lt;0.4</td>
</tr>
<tr>
<td>Related</td>
<td>&gt;0.4 &lt;0.6</td>
</tr>
<tr>
<td>Strong</td>
<td>&gt;0.6 &lt;0.8</td>
</tr>
<tr>
<td>Very Strong</td>
<td>&gt;0.8 ≤1.0</td>
</tr>
</tbody>
</table>

This form of learning takes place only after the first set of documents have been retrieved. Once a set of documents corresponding to a series of concepts are retrieved, the learning function can be activated. Documents retrieved from different concepts are compared. The level of relationship is derived in terms of common words and a numerical value is computed. This computed value is then compared with the current level of relationship, leading to a confirmation of the present value if it is close enough to the computed value. It may also cause a new level of relationship between the two concepts to be suggested if the computed value is in a different range.

6.6 Some Implementation Issues

IACP runs in a Hypercard environment. To meet the necessary level of functionality, Hypercard was augmented with several
external code modules known as external commands. The user will not, however, be aware of this fact. In addition to Hypercard an Apple research product called the Logic Manager was used. Logic Manager operates as a set of 6 external commands used within Hypercard scripts. The retrieval component and many of user interface functions also take advantage of external commands written in C and called from within Hypertalk scripts. IACP was developed on a Macintosh Classic II with 4 Megabytes of memory. To transfer the geography documents into the document base, an optical scanning device using Omni Page optical scanning software was used. IACP occupies two high density floppy diskettes. When it operates, a hard disk or another high capacity storage medium is required for storage of the retrieved material. The document base also requires space, depending on its current volume and future expansion required.

6.6.1 Hypercard

Hypercard was introduced in August 1987 by Apple Computer Inc. A brief description of Hypercard is contained in Appendix C.

6.6.2 The Logic Manager

The Logic Manager is a small inference engine, available from the Apple Research Group. It is briefly described in Appendix D.
6.7 Application of IACP to a Geography Course Preparation

Geography was chosen as the domain to test the ideas presented in this thesis, for which there is at least one precedent. SCHOLAR, [Carbonell, 70] one of the first ITS, was built to teach the geography of South America. A number of reasons influenced our decision to choose geography as the experimental domain. The main reasons were:

- Non-algorithmic nature of geography
- Conceptual structure of geography
- Previous other experience with the same domain in the application of computers to education.

In the following paragraphs we will briefly discuss these factors, referencing the available literature where possible.

Geography is a social science. The knowledge in geography, like other social sciences, is not algorithmic. In domains such as mathematics or physics the knowledge is mostly algorithmic. Such a domain would require a different treatment, in which case the idea of CR models might not be applicable.

Geography is about the environment and the way it embraces man, land and the relationships between them [Fielding, 74]. Concepts and how they relate to one another seems to be a natural way of structuring knowledge in geography. This may be the reason for previous studies along these lines. Fielding [74], in a book titled "Geography as a Social Science" has organised the material
in the book around the main concepts of the topic. One of the objectives of the book stated by the author was to identify the main concepts used by geographers in that field. Shortle [1974], in a study of the conceptual structure of geography, discusses how the concepts in geography are related to one another. The idea of identifying concepts in geography, both abstract and low-level, has been used by geography education researchers [e.g. Stringer, 75; James 65; Hartshorne, 59]. These studies make us more confident of our intention to encapsulate the structure of geography as concepts and relationships between them. Similarities with other social sciences should make our approach applicable to them.

Geography has been used for research on the use of computers in education before. This has been in the form of building CAI courses in geography (e.g. Fielding, 74) and for testing ideas in the field of ITS (e.g. Carbonell, 70). This again was another factor influencing our selection of geography for testing our approach based on our model of domain structure which we have referred to as the CR model.

After choosing geography as the domain, an expert familiar with subject matter familiarity was sought. As expected, finding a suitable expert took some time. The first expert chosen was an academician with many years of experience. It took some time to arrange the first meeting with the expert. The initial meeting went very well as the expert appreciated the ideas and commented that a tool like IACP would be of great help to his department. The audience would also have been a very suitable as they would have been well advanced in their abstract thinking stage of mental
development. A tight schedule of both executive and academic duties of the expert caused a long gap between the first and the next meetings. After the second meeting it was decided that it was safer to find an expert who would be more available. A geography teacher with several years of experience was the next candidate, and although this meant a shift in the audience addressed by our prototype from university students to high school students, there was little choice if the construction of the prototype was to be completed in reasonable time.

Naturally, there was no CR network of the domain available, so producing one was the first task. The CR network had to be built from scratch, and because this was being done for the first time, several problems arose.

Our approach to knowledge acquisition requires the structure of the domain to be elicited before the domain knowledge is acquired. The second step, which is the acquisition of domain specific knowledge, is based on the framework built in the first step. Building a CR model requires both of these steps to be followed.

In the first step, the CR model, without the key knowledge required for student modelling, is constructed. In the second step key knowledge is acquired from the expert and the model is completed. The expert co-operating in the first step may be different from the one co-operating in the second step, but in this case they were one and the same person. In the initial meeting, the aim of the project, and the approach, were explained to the expert.
The approach was new and there was no previous experience to refer to. This caused certain doubts in the mind of the expert. It took some time and explanation to gain the confidence of the expert in the project. The early meetings with the expert took time and were not particularly fruitful. This was due to the expert's misunderstanding of the terminology, and the approach itself, arising from a lack of similar experience. Other people familiar with the project had to be involved during the initial meetings. Once this initial stage passed, the interviews went smoothly and only problems commonly encountered in knowledge acquisition were faced and solved. At times, we had to go back to previous stages in structure elicitation and revise the results obtained in those earlier stages. This happened even in the testing stage, when we had to go back to the initial stage. Once the expert was convinced of the worth of the approach, we started working toward selecting a topic within the broad domain of geography. This would probably not happen in an actual non-research application of IACP. A teacher using IACP would start using it with a topic already in her mind. We initially had the physical geography of an area (a continent) in mind for the prototype. The expert teaches "development" in the context of geography to St. Mary's College students, so she suggested that for the topic, as it would provide the chance for her students to use the system.

In several meetings with the expert, which followed the methodology presented in Chapter 3, a CR model of development was constructed. A portion of this network was shown in Figure 6.1. The complete network, as it is graphically represented on cards in IACP, is included in Appendix A. The initial version took
several revisions through interviews with the expert to reach the stage shown in the appendix. The system was completed and the network was entered into IACP and tested. On reviewing the results of the testing, the information units of the concepts had to be revised.

After revisions, the system was tested again. The hypertext was successfully constructed. The student modelling component was checked and appropriate changes made. The system was then tested at St. Mary's College in Wollongong. Eleven subjects were chosen from students with various backgrounds. Some of the students had taken the course, with the expert as their teacher, and some were going to take it in the next year. To make the time spent by the students shorter, only three concepts were selected for the purpose of testing. These were:

- Distribution of Wealth
- Government Policies
- Use of Technology.

The models constructed of students are summarised in Table 6.2. The results show that IACP has constructed an acceptable model of the students.

The second column in table 6.2 indicates whether the student has taken the course on development before or not. The third column shows the number of concepts that the student was able to identify. The students who had taken the course have been able to identify the concepts much better those who have not. The students
who had taken the course identified all the concepts except R who missed one. Those who had not taken the course had difficulty identifying the concepts except C who identified all the concepts.

TABLE 6.2 Summary of Student Models Constructed by IACP

<table>
<thead>
<tr>
<th>Student Name</th>
<th>Taken Course</th>
<th>Identified Concepts</th>
<th>Identified Relations</th>
<th>Score</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>YES</td>
<td>3/3</td>
<td>2/2</td>
<td>80,60,46</td>
<td>K,K,PK</td>
</tr>
<tr>
<td>L1</td>
<td>YES</td>
<td>3/3</td>
<td>2/2</td>
<td>80,100,66</td>
<td>K,K,K</td>
</tr>
<tr>
<td>R</td>
<td>YES</td>
<td>2/3</td>
<td>1/2</td>
<td>65,70,42</td>
<td>K,K,PK</td>
</tr>
<tr>
<td>Y</td>
<td>YES</td>
<td>3/3</td>
<td>0/2</td>
<td>40,20,42</td>
<td>PK,NK,PK</td>
</tr>
<tr>
<td>L2</td>
<td>YES</td>
<td>3/3</td>
<td>2/2</td>
<td>80,65,35</td>
<td>K,K,NN</td>
</tr>
<tr>
<td>C</td>
<td>NO</td>
<td>3/3</td>
<td>1/2</td>
<td>85,55,17</td>
<td>K,PK,NN</td>
</tr>
<tr>
<td>M1</td>
<td>NO</td>
<td>0/3</td>
<td>0/2</td>
<td>35,15,20</td>
<td>NN,NN,NN</td>
</tr>
<tr>
<td>M2</td>
<td>NO</td>
<td>2/3</td>
<td>0/2</td>
<td>60,5,12</td>
<td>PK,NN,NN</td>
</tr>
<tr>
<td>J</td>
<td>NO</td>
<td>1/3</td>
<td>0/2</td>
<td>55,0,5</td>
<td>PK,NN,NN</td>
</tr>
</tbody>
</table>

K: Known  PK: Partially Known  NN: Not Known
C1: Distribution of Wealth  C2: Government Policies  C3: Use of Technology

The third column in the table shows the number of relationships each student was able to identify. Again, those who did the course before had less difficulty with the relationships. In fact, only R missed one relation and Y could not identify any. This
was not unexpected as the students were not at the same level in each group. Comparing the performance of R and Y, we can see that they had more difficulty identifying the relationships than the concepts themselves. As the table suggests, students who had not done the course on development could not identify the relationships between concepts, except for one case.

The students' performance in showing how much they understand the domain structure in terms of the concepts and relationships shows that those who had taken the course before were more successful in this respect. This is in agreement with the assumptions we made in Chapter 5 concerning the relationship between the student understanding of the structure of the domain and their understanding of the domain, which is used as one source of evidence in constructing a model of the student. This also finds support in cognitive psychology research, as discussed in Chapter 3. In short, a student who understands a subject should also have an understanding of the structure of it, which supports our use of the CR model as the medium for modelling the student.

Column five shows the total score of each student for each of the concepts. The total score is the sum of scores obtained from answering the questions from the key knowledge acquired from the expert, filling the blanks, and identifying the concepts and relationships. A score of 65 or above is taken to mean that the student knows that concept. This leaves room for correct student answers not matching the student modeller's version of the correct answer or the synonyms recorded for it, as well as the unrelated questions generated from retrieved text. The text retrieved may
not always be related. The latter could have been treated by giving a smaller weight to this type of question.

The last column in the table shows the student's status on each of the concepts. Again, looking at the status of the students having taken the course, we can see that except for Y they have done well. Comparing that with the performance of the students who had not taken the course we see that this picture matches their background knowledge.

The overlay model showing which concepts are known or not, and by which students, can be put to different uses, depending on the aims of the teacher. The system can be used to generate a hypertext for each student which excludes the concepts they know. It can also be used for guiding the student through the whole hypertext when they try to go to a node which has prerequisite knowledge which the system knows that the student does not know. Concepts partially known can be managed the teacher's preferred strategy. The student models as we discussed them in Chapter 5 can potentially be used in an ITS.

Students were asked to comment on the system. The suggestions made by the students can be summarised as follows:

1. They preferred a list of choices to choose from when answering questions
2. They thought material to go through for finding answers to questions was necessary.
The first suggestion is already fulfilled by providing lists to choose from where appropriate. The second suggestion is fulfilled with the hypertext which can be used by the students for becoming more familiar with the domain. This suggestion supports the idea of the making hypertext available for the students to explore in what is known as exploratory learning. A sample of student written responses in writing is included in Appendix B.

6.7.1 Some Details of the Test

The student modeller makes its decision based on three sources as discussed in Chapter 5. These are the student answers to questions from text, her answers to questions from the knowledge base and the student's understanding of the domain structure in terms of its concepts and their relationships.

The process of student modelling begins with questions that are asked for bookkeeping purposes. The student is asked to enter her name and an area is set up under that name for storing the model constructed. The next question asks the student whether she is ready. The student can choose to continue or quit at this stage. The next display is a message telling the student that she is going to be filling in blanks. Then a set of questions generated from text for one of the concepts is asked. The three concepts selected in this case had the same difficulty level, and the questioning started with the concept of "Distribution of wealth" as follows:

The term "north Atlantic nations" is used to refer to countries of western Europe and North America.
The answer to this question should be "rich" or a synonym such as wealthy. We realise that the student may come up with a correct answer different from the preferred word and its synonyms. This is taken into account by considering 50% in this part of the testing (filling the blanks) to be an acceptable score when making a decision about the student's understanding of the concept. This includes account of the possibility of questions being not directly related to the concept similar to the next question which is:

Governments that were inclined towards socialism were more directly involved in -------- their national economies than were those of the more capitalist philosophy.

The answer to this question is, "developing". A teacher might want to exclude such questions, in which case the scoring would become slightly different. Other questions asked in relation to this concept are:

Debts on past aid have threatened to stem the flow of new aid from -------- to poor countries.

The reason families in the -------- world choose to have large families are complex.

The bulk of -------- are found in the neglected rural areas where hundreds of millions of people scrape a meagre living through agriculture.
Two thirds of the world's children live in the countries and for most of them malnutrition is a fact of existence.

An enormous and widening gap in well-being exists between the minority of the world's population in the developed industrialised nations and the poor majority in the developing countries.

If they do not get protein food they will eventually die. (not quite relevant)

Questions from the knowledge base follow questions generated from text. For the same concept the questions asked were:

The majority of people in developing countries are still well off but live in rural areas. (true/false)

In a developing country there is a minority of very wealthy people with a commercial life style. (true/false)

These questions are true/false type questions. After several questions are asked, the student is asked to identify the concept she was questioned on. A list of concepts is displayed for the student to select from. An example concept list is shown in Figure 5.11.

The questioning continues in the same manner for the next concept, which is "use of technology". The student goes through
filling the blanks and questions generated from the knowledge base which could be of different types. They could be similar to those presented in the previous paragraphs or they could be short answer type questions, answers to which could consist of one or more words, such as:

What is the most suitable form of technology for developing countries?

the answer to which is, "labour intensive". If the student identifies the concept "Use of Technology" after answering the questions, then the system checks for direct relationships between the concepts identified. In this case there is no direct relationship and no questions are asked in this respect. The next concept is "government policies". The same type of questioning is done on this concept. If the student identifies this concept following the questions, then checking for possible relationships is commenced. This time there are two relationships between the latest identified concept and the previously identified concepts. In this case, the following questions were asked:

You have identified concepts, "government policies" and "distribution of wealth". Can you identify the relationship between them?

The student must then select the appropriate relationship from a list similar to the list of concepts. The relationship in this case is, "regulates". There is another relationship to be identified. This is done in similar fashion by asking the question,
You have identified concepts "government policies" and "use of technology". Can you identify the relationship between them?

The relationship in this case is "greatly affect". These relationships are the ones acquired from the expert while constructing the CR model. A student's score is the total score obtained from the three types of questions.
7. Summary and Conclusion

7.1 Summary

An authoring shell is an ideal general purpose tool for constructing intelligent tutoring systems. Although such a tool is not currently possible, we have proposed it as an ideal tool, towards which research should be directed. We have represented an authoring shell as a pyramid with a changeable tip (Figure 1.2). This brings about a high level of separation between content and code. This matches the trend as depicted in Figure 1.1.

The top part of the pyramid shown in Figure 1.2 contains domain specific knowledge of different forms. The bottom part contains tools and procedures for constructing the top part and for manipulating the knowledge contained therein for the purpose of teaching. Procedures for transferring knowledge from the sources of knowledge (e.g. human experts, books, videos), structuring of this knowledge and transferring it to the learner in an individualised manner should be included in the bottom part of the pyramid.

Knowledge representation is an important topic of research in artificial intelligence. It is also a central issue in intelligent computer assisted instruction. Different schemes of knowledge representation are needed for the various components of an ITS (e.g. student model, KB, Tutoring). We discussed earlier that different methods of knowledge representation must be used in combination to represent domain knowledge.
Starting in the 1960's with the introduction of Piaget's mental development theory, researchers have devoted their attention to how human learning takes place and cognitive structures are formed. As a tool, concept mapping has been used to measure changes in learners' cognitive structure. Concept mapping is however yet another form of semantic networks and a formalism for representing domain specific knowledge. In looking at knowledge representation and education in geography we found studies that have actually used a graphical knowledge representation tool called concept maps in classroom situations. Ghaye [84] and Robinson [86] use concept maps to assess student learning progress with high degree of success. There are other studies which have used concept mapping for the same purpose in other domains. The way concept maps are used is to ask students to construct a concept map of what they have been taught in class, or a passage, or a page of test they have been given for this purpose. Concept maps are very much yet another version of semantic networks suitable for representing domain knowledge. Concept maps would be a useful tool for assessing students. They are much more detailed than the model we needed. We needed an effective formalism for representing the structure of a domain rather than detailed domain knowledge.

We have developed a model for representing the structure of a domain in terms of the key concepts and how these concepts are related to one another. We have called this model the concept relationship model (CR model). This model fits our needs. Based on the structure of the domain graphically represented using CR
networks, retrieval, organisation, sequencing of resources (including text and/or graphics, video, etc.), becomes possible. This has been shown by constructing a CR network of the topic of development in the domain of geography and using it to structure course material for courseware construction. Further, the CR network depicting the structure of the domain, and the relationships among the concepts therein, has been used for assessing the student understanding of the subject matter, both independently of content matter and using the retrieved material. This is an effective way of assessing students [Bruner, 1960; Ghaye, 1984; Robinson, 1986].

The concept relationship model consists of the abstract concepts forming the domain, the relationships among the concepts, and information which puts each concept and relationship in the position where it belongs in this structure. Each node in a concept relationship network represents clusters of information which can be used for teaching that concept. Each of the concepts and the relationships has an information unit associated with it. An information unit (descriptor) of a concept holds information on the naming used, difficulty level, hierarchy level, synonyms, related concepts, sets of keywords and weights associated with the keywords, context lines and key knowledge associated with that concept. Each piece of information serves a specific purpose. Other information such as knowledge representation forms and media information is also recorded for later use. The information unit of a relationship contains a level of strength and keywords in common. Other bookkeeping information is also kept in the information units of both concepts and relationships. The CR model
is constructed through interviews with an expert. The expert is expected to have subject matter knowledge. Because CR networks represent the structure of the domain, and contain information about the material related to each concept, as well as the position of each concept within the domain structure, we were able to use them as a basis for automatic conversion of text into hypertext. Within the same context, other media may be used, in which case automatic hypermedia linking will be possible. A problem which may arise in such a case is that of indexing, as pointed out by Schank [93].

We view knowledge acquisition as consisting of a domain structure elicitation and a domain knowledge acquisition component. During domain structure elicitation, a CR network of the domain is constructed through interviews with an expert. We have developed and used a methodology for structure elicitation. A life cycle, along with tools and guidelines for following the life cycle, have been presented.

We have constructed an intelligent assistant for courseware preparation (IACP) based on the CR model. This tool is one with potential for use in the authoring shell model we are proposing here. The intelligent assistant for courseware preparation (IACP) retrieves, structures, and links material for use in a course. It also creates a student model. Intelligent links are possible by combining the two outputs. In IACP, the CR model is a basis of generating a system and also acts as an intelligent layer above the hypertext automatically generated.
The hypertext generation subsystem is one of the components of the IACP architecture. Several hypertext and hypermedia systems have been constructed and used, mostly for instructional purposes. Some of the most influential systems are:

NoteCards
KMS
Intermedia

However, little prior work has been done on automatic, semantic linking of text or other media. In all the systems mentioned above, the user manually creates the links. Based on the CR model, we have linked the documents retrieved for use in courseware construction automatically. The resulting hypertext can then be browsed by the teacher for screening (selection). The students can also explore the hypertext in exploratory learning mode if the developer of the courseware chooses to let them. Based on the information contained in the CR network, and the model of the student also constructed by the system, the hypertext links can be made intelligent, so that, if a student chooses to jump to a node for which, based on the student model, she does not know the prerequisites, then she is asked to go to the prerequisite nodes before going to the desired node. Two general types of links are used in IACP. A document level link is a lower level link connecting the documents in the hypertext. These links may themselves be internal or external. Internal link refers to a link between documents of the same concept. An external link refers to a link between a document of one concept to the documents of another concept. The high level links are the links at the CR
network level, linking the concepts. Intelligent linking is possible at both levels.

One of the main components of an ITS that we identified in Chapter 2 is the student modelling component. Student models are a merit of ITSs. Student modelling is well researched. We studied the different methods of student modelling and highlighted the main stream of research in this area in chapter 5. The authoring shell we have proposed as the ideal tool for the construction of ITSs provides for student modelling. The intelligent assistant for courseware preparation (IACP) that we have constructed generates a student modelling component which can be used as the student modelling component of the ITS being constructed. IACP also supports conventional CAI environments. Conventional CAI environments do not provide for student modelling. IACP generates a student modelling component with every collection of material generated for a course. If the material is generated for use in a CAI environment then the student models can be used for generating several versions of the same course each of which is suitable for a group of students in the same class. A class may be grouped according to the concepts they all know. This grouping can be approximate and concepts known by the group will not be included in the lesson customised for that particular group.

IACP uses both the CR model of the domain and the material retrieved to construct a student modelling module. The student model tests the student understanding of the domain through:

The key knowledge contained in the CR model
Questions are generated from the knowledge base and presented to the student. The student's responses are checked against answers to the questions also generated from the knowledge base. Clozes, which are sentences with missing words, are presented for the student to fill in the blanks. Finally, the student ability to identify concepts and relationships between them is assessed. The student is given a score at each stage and the weighted score sum is used to reach a decision on whether or not the student knows a concept.

The information collected by the student modelling component can serve two purposes. The main purpose and the aim of this study is to use it in the target intelligent tutoring system as a basis for presentation of course material. The second interesting use of the student model is to make the hypertext links generated intelligent, as discussed in Chapter 6, and make the hypertext created readily usable for instruction. It is worth noting that, by comparing hypertext with instructional design processes, Jonassen [Jonassen, 1992] has shown that hypertext systems are powerful tools for both instructional development and delivery.

IACP is now operational and can be used for courseware preparation in various domains. We have used it for this purpose in the domain of geography. A CR model of the topic of "development" was constructed through interviews with an expert on the topic. The CR model was then fed to the system. IACP was then used for constructing several customised CR networks and for retrieving,
structuring, and linking the material. IACP was then tested for validity of the student models it generates at St. Mary’s College and favourable results were obtained both in terms of the models generated and student’s comments. IACP is an academic prototype and has helped us make the step forward towards the authoring shells we are proposing. It is a tool that reduces the amount of time spent constructing computer assisted instruction programs. Educators prefer to be the producers of their courseware. IACP makes rapid production of courseware possible for teachers who should be the actual producers of courseware.

7.2 Conclusion

In the last two decades, researchers in intelligent computer assisted instruction have applied various techniques from artificial intelligence to the development of intelligent tutoring systems. The area of intelligent tutoring systems, however, suffers from the lack of the powerful tools required. As a result, the intelligent tutoring systems constructed up to now are usually constructed by computer science researchers for testing research ideas. Their domain is limited and the cost of building then is high.

Conventional computer assisted instruction enjoys tools such as authoring languages and authoring systems. Authoring systems make it possible for non-programmers to construct courseware. This may explain why conventional computer assisted instruction systems are often used in schools. An area of artificial intelligence which has impinged on intelligent tutoring systems is expert systems. Researchers in ITS have used expert systems as a
basis for constructing ITSs. An example of such efforts is GUIDON, built on the top of the MYCIN medical expert system, for teaching in the same domain. General expert system shells have also been tried for application in education. Expert system shells in a way resemble authoring systems. We believe that it is possible to conceive a tool similar to an expert system shell for constructing ITSs. We have represented this idea in the form of a pyramid which consists of two parts as depicted in Figure 1.1. The bottom part of the pyramid is to be a collection of tools and the top part to be a collection of domain knowledge in various forms. With the current state of technology, such an ideal tool cannot be constructed. It is believed that it will probably take another decade before general purpose tools for constructing ITSs come into existence. We believe that with the ideal model which we have called authoring shells (explained in chapter 1 of this thesis) in mind we should work on constructing tools to be integrated into this model. We have done so by constructing an intelligent assistant for courseware development.

The intelligent assistant (IACP) that we have constructed reduces the time spent on constructing an ITS or a conventional CAI system by retrieving, linking and organising material and building a student modelling component to go along with the retrieved material. This tool is one of the tools needed in the bottom of the ideal authoring shells pyramid. By providing a hypertext environment using the material retrieved, a learning environment is made readily available by IACP. The student modelling component generated by IACP can be used to further
enhance this environment to a semi-guided discovery environment. This adds to the usability of IACP.

Knowledge representation is a central issue in artificial intelligence. ITSs, however, have their own knowledge representation requirements. In ITSs we need the knowledge of solving problems, and we need the knowledge of how to teach this material. The knowledge has to come in a variety of forms, suitable for presenting to learners and for manipulation by the machine. We believe that both the knowledge of problem solving and the knowledge to be presented for teaching should come in a variety of forms suited to each. As it is more appropriate to present an interview in the form of a video clipping and a table in printed form, different knowledge representation formalisms are suitable for representing different types of knowledge used in an ITS. One type of knowledge may be formalised using frames, and another using first order logic, or semantic networks, depending on the type of knowledge.

It is claimed by cognitive psychology researchers that things are stored in long term memory through a structure similar to a semantic network. A similar structure called concept maps is used in cognitive psychology research to assess learners. We believe that a higher layer structure for organising domain knowledge is necessary. The concept relationship model explained in Chapter 3 is intended to be a high level structure of the domain of interest. The CR model was developed in this study and used as a basis for constructing IACP. Because the CR model represents the structure of the domain it can be used for structuring the material for
teaching a subject. We have used them as a scaffolding structure for both retrieval and organisation of subject matter. The CR model is used by IACP for automatically generating hypertext paths through retrieved material. This makes then a useful general purpose medium for automatic hypertext generation.

The multi-layered hypertext model we have introduced here is an intelligent network that is used for navigating through a document space. This higher level network, which is a CR network, is not to be mistaken with the so-called web sometimes used as a graphical representation in some hypertext systems (e.g.. NoteCards). A web actually mirrors the links between the documents in a hypertext. It is a picture of the document level links. The CR network on the other hand, is metaknowledge of the domain depicting its structure. The CR network makes a new way of linking possible with features making it totally different from a web as follows[Sarrafzadeh and O'Brien, 93]:

- The CR network is independent of the document base it links together
- The CR network can be used for automatically linking a set of documents
- Intelligent linking is possible, making the hypertext generated more suited to educational purposes
- The CR network links clusters of documents
- Each document is linked to its corresponding concept node
- Each cluster of documents is connected to its concept node
Two clusters are linked through the CR network (This is in addition to the lower document level links which may exist between documents).

The CR network can be enhanced by modifications to the strength of its relationships as a result of learning.

The CR networks are customised to match the user's ideas.

Document mark up is not necessary.

New documents can easily be added to the document collection.

The CR model is a medium for the purpose of courseware preparation, and can be useful in general hypertext as well as other applications such as network based selection of material for various purposes, including educational purposes.

A CR model of a domain is constructed through knowledge acquisition. We view knowledge acquisition as consisting of two main stages of structure elicitation and acquisition of knowledge from its sources. This two staged approach to knowledge acquisition is not only appealing to ITS construction but can be useful in general knowledge acquisition [Sarrafzadeh and Booth, 93]. The model is the basis of the intelligent assistant and central to the idea of authoring shells. The general problems of knowledge representation related to the acquisition of knowledge from the expert still exist and the result of IACP's operation is very much dependent upon the CR networks constructed. The CR model once constructed can be used as a basis for acquisition of problem solving knowledge. This task is to be automated in the ideal authoring shells.
The idea of automatic linking and intelligent links made possible through the CR model can lead to the development of semi-guided discovery learning environment with little effort on the part of the end user. To make this possible, CR models of domains of interest must be constructed by knowledge engineers and experts who should be familiar with subject matter. Care in construction of the model is necessary as it is the backbone of the whole system.

The CR model we believe is a good basis for automatic hypertext and hypermedia generation. Not only that, but it addresses some of the previous problems of these systems, because the CR model (customised) acts as a higher layer of intelligent links above the document level. This ameliorates the main problem of "disorientation" common to hypertext systems.

We used IACP for building a sample system in the area of development in the general context of geography. The material retrieved, linked and organised was a useful learning environment on its own, as well as being suitable for use in a course. The need for such an environment was reflected in the comments of the students who participated in the tests conducted at St. Mary's College to assess the effectiveness of the student modelling component. The student models constructed reflected the students' level of knowledge. The students were two groups, one which had taken the course before, and one which had not. Both groups believed that an environment such as the one generated by IACP would enhance their performance.
Finally we believe that the contributions made by this work, (e.g. automatic hypertext generation, domains structure elicitation, courseware material preparation, student modelling and intelligent linking) are useful steps in the direction of ideal authoring shells. They are also useful in the area of hypertext systems. Automatic generation of semi guided discovery learning environments is also possible using our approach. A variety of media can be employed in this context, as long as the problem of indexing is solved. Once ways are found to index a collection of information, it can be incorporated in IACP. This makes IACP a potential multimedia system. With the automatic linking capability of IACP, addition of other media would also make the system a hypermedia system.


Benest, I.D., Computer Assisted Learning Using the Dynamic Book,


Bush, V., As We May Think, Atlantic Monthly, July 1945, PP. 101-108.


Carr, B., Wusor-II: A Computer-Aided Instruction Program With Student Modelling Capabilities, AI Lab Memo 417 (Logo Memo 45), MIT, Cambridge, Massachusetts.


Hartshore, R., Perspective on the Nature of Geography, Chicago, 1959.
Hauptmann, A.g., Green, B.F., A Comparison of Command, Menu-Selection, and Natural Language Computer Programs, Behaviour and Info. Tech. 2(2), pp163-178 1983


Jones, R., Automatic Document Content Analysis: The AIDA Project, Library HI-Tech, Fall 19991.


Merrill, M. D., and ID2 team, Instructional Transaction Theory: Knowledge Relationships, Educational Technology 31(6), 1991, pp. 7-12.


Shneiderman, B., User Interfaces, Video Course, University of Maryland, USA, 1989.


Williams, G., Hypercard, Byte 1987(Dec.), PP. 109-117.


APPENDIX A CR Network of Development

- regulating distribution of wealth
- government policies greatly effects standard of living and development of manufacturing
- use of technology determines development of manufacturing
- determines world location
- climate water available 1 affects soil fertility 1
- determines types of vegetation 2
climate 4

vegetation

determines types of

soil fertility

climate 2

types of agriculture 2

soil fertility 3

determines

types of agriculture

landform types

facilitates

transportation

facilitates(1)

is essential for

development of manufacturing

amount of water available 2
Government policies greatly effects development of manufacturing.

- Motivation
- Production of surplus food
- Availability of water
- Use of technology
- Transport
- Imposition of import/export
- Amount of water available
- Capital

Amount of water available is essential for:

- Climate
- Vegetation
- Types of agriculture

Development of manufacturing creates jobs.
Determines composition of import/export

Trade balance of import/export determines (1) world loc

Guarantees capital

Facilitates/improves development of manufacturing

Creates jobs
Manufacturing enhances/facilitates availability of minerals; it also contributes motivation. Manufacturing is a precondition for production of surplus food. The standard of living improves with an efficient distribution of wealth. This, in turn, enhances role of women in the economy and reduces population low growth.
APPENDIX B: A Sample of Students' Comments
APPENDIX C: Hypercard

Hypercard is a multimedia presentation program. It is perhaps also the most readily available hypertext tool. Because Hypercard is included with all Macintosh computers, many people have access to it. Hypercard with its index card metaphor makes manual linking of text, graphics, sound and video possible. Hypercard provides its user with an English-like language which is, in a sense, an object oriented language. Hypertext has some limitations and is not fast. The limitations, and the problem of speed, can be overcome to an extent through the use of external commands (XCMDs). XCMDs are programs written in a high level language which can be called from within Hypercard scripts.

Hypercard is based on the index card metaphor. A card in Hypercard may be used to show text and/or graphics. A card may contain other objects such as buttons and fields. The information visible on a card may belong to a single card, a group of cards or an entire stack. This is done through backgrounds. Backgrounds contain shared information. Buttons are used for causing an action to take place when they are clicked. They come in different shapes and may or may not be visible. Text and other information is usually stored in fields. Variables are also used for holding information. Fields are different in that they store information permanently. So, a card may have buttons or fields on it. Cards may share information contained in the background they belong to. Animations, sound or video can be attached to a card. A collection of cards forms a stack.
A special stack called the home stack is the "central station" of Hypercard.

In Hypercard an object may be a button or a field, a card, a background or a stack. Programming Hypercard is different from languages like C, Pascal or Basic. In Hypertalk, the language of Hypercard, each object is controlled by a script and objects communicate by sending messages. Hypertalk scripts are event based. They execute when a specific event occurs. An event may be generated internally (e.g. idle) or may be generated by the user (e.g. mouseDown, TabKey). Each event is passed as a message through a hierarchy. When a message is generated it travels through this hierarchy until it finds a handler (script) that handles it. These message handlers correspond to so-called methods in object-oriented languages. A button or a field is a first receiver of messages. The current card, the current background, the current stack, the home stack and finally Hypercard itself, form the hierarchy through which messages travel until a handler is found or an error message is generated. Objects can send messages to a specific destination.

Hypercard provides several commands. The user of Hypercard uses these commands to write scripts for the objects created. New commands are added to Hypertalk by copying them using a resource editor such as ResEdit. Several XCMDs were added to Hypercard for the implementation of IACP.
APPENDIX D: The Logic Manager

The Logic Manager (LM) is a small inference engine capable of providing advanced programming ("intelligent") capabilities to Hypercard and other Macintosh applications. It does this by providing databases for storing facts and sets of rules in order to deduce answers, satisfy constraints, perform search, etc. Logic Manager includes most of a Prolog interpreter. It thus provides general logic programming capabilities. It is meant to be incorporated into applications for experimenting with the AI technology.

The reason Logic Manager was chosen was that it is available both as an MPW object library, and as a Hypercard external command. LM itself was implemented in MPW C. The problem it has is that it does not provide a development environment. This problem has been solved through a conversion utility that converts Prolog code written in the Edinburgh dialect of Prolog to LM syntax.

The LM is not an inference engine meant for use by end users. "The LM might be used by any application developer, but because of the limited state of documentation, development environment, and support, we think that this early version should be used by developers who are 1) Macintosh literate; 2) AI literate, preferably skilled in Prolog; and 3) oriented more toward research prototypes ..."[LM 91a]. Our initial intention was to use a new version of LPA Prolog which allows for the use of Prolog from within Hypercard through apple events. Logic Manager was however preferred as it is

33
both much smaller in size (~ 50K object code) and does the job well and fast.

Logic Manager can be used from Hypercard scripts. This is achieved through an External Command (XCMD) for Hypercard. In a way, this makes LM an extension to the Hypertalk language. The XCMD called 'LmMgr' may be called from Hypertalk scripts using the following syntax:

\[ \text{LmMgr <cmd-code> [ <additional-parameter> ]} \]

where,

\[ <\text{cmd-code}> ::= \{ 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \} \]

This command code is required.

A command code of 0 is used to initialise the logic manager. This has to be done prior to any operation through the LM. A command code of 1 causes the LM to quit. This must be the last operation done in LM. Command codes 2 and 3 are used to set up memory for LM to operate. Command code 4 is used for asserting facts or rules into the knowledge base of the LM. Command code 5 must be used to initiate queries. It must precede command code 6 which is used to make a query to the knowledge base. Command codes 2, 3, 4 and 6 require an additional parameter. In the case of command code 2 and 3 this additional parameter is the size and other specifics of the area in memory which is required. Command code 4 is followed by the rule or fact which is to be asserted into the knowledge base. Command code 6 is followed by the query [LM, 91b].
The LM must first be installed by copying the LmMgr XCMD into the home stack, Hypercard or a specific stack using either ResEdit or the ResCopy. We have installed the LM into Hypercard and from there it is used by IACP. Before the LM engine is started, the file called LMCode must be loaded. This was explained in the operation of IACP.