TECFA

Technologies de Formation et Apprentissage
Educational Technology Unit

The MEMOLAB Project

Current Status, June 91.

P. Dillenbourg, M. Hilario
P. Mendelsohn and D. Schneider

TECFA DOCUMENT 91-5

Project No 4023-27013
Research funded by the Swiss NFP23 Program,
"Artificial Intelligence and Robotics"

UNIVERSITÉ DE GENÈVE
Faculté de Psychologie et des Sciences de l'Education

Abstract

MEMOLAB is a learning environment for acquiring basic skills in experimentation methodology for human sciences. During the first year of research, we designed the *cognitive architecture* of MEMOLAB.This architecture translates into computational terms the neo-piagetian theory of development proposed by R. Case.Each stage of learning in MEMOLAB is associated with a set of interaction patterns, supported by a specific interface. The dynamic link between stages makes use of the computer as a tool for reflection.

In parallel, we worked out MEMOLAB's *computational architecture*. The LAB interface allows learners to build an experiment. The SIMULATION produces the results of this experiment. For pedagogical and knowledge engineering reasons, we have chosen CASE-BASED REASONING techniques to implement this simulation. These techniques have been adapted to fit the peculiarities of the process to be simulated. We developed a HYPERTEXT system, integrated into MEMOLAB, that provides an 'Encyclopaedia of Memory' and other on-line assistance. The multitude of agents within the system led us to pay more attention to control issues. We decided to use a BLACKBOARD system to address those issues. We built an inference engine, integrated in the object-oriented approach of MEMOLAB. This engine will support the blackboard component. We have specified four COACHING agents, three content-specific tutors and a domain-independent coach which guarantees that the learning environment provides optimal learning conditions for a given student, at a given time.

Contents

1. Artificial Intelligence and Education: State of the Art

Research in "AI and Education" is based on the computational study of learning and teaching processes. The design of intelligent learning environments (ILE) raises several challenges for AI researchers, both at the technical and epistemological levels. The reciprocity of the relationship between AI and ILE has been illustrated by the work of W. Clancey (1983). His attempts to use the well-known expert system Mycin as a training tool showed the limitations of early production system architectures. He demonstrated the necessity of explicitly representing problem solving strategies and thereby prefigured the second generation of expert systems (Steels, 1990).

Nowadays, several AI techniques are tested or developed through the implementation of ILEs. Recent developments on belief systems are explored in order to bypass the difficult problem of student modelling (inferring the student's misconceptions from her behavior on the interface) (Self, 1991). Dynamic planning methods, especially blackboard systems, are used to allow the tutor component to adapt its didactic plans to the student behavior (Murray, 1989). Among other techniques, we point out the use of classifier systems (Hron, 1990), case-based reasoning techniques (Bloch and Farrell, 1988), explanation-based learning (Costa and Urbano, in press) and inductive learning (Dillenbourg, 1989) methods, neural networks (Mengel and Lively, 1991), distributed problem solving (Brazdil, in press). Some of these techniques are used in MEMOLAB, the system that we are developing. This report describes how pedagogical constraints have shaped the particular adaptation of these techniques to the MEMOLAB's functions.

However, the scope of artificial intelligence goes well beyond an inventory of techniques. AI is concerned with epistemological issues as well. Those issues are critical when designing an ILE (Wenger, 1987). The 'symbol grounding' crisis (Harnad, 1990), that has shaken the symbolic paradigm, has also affected our discipline. Focus has moved away from internal and mental representation towards external representations and social interactions. The recognition of the social grounds of individual knowledge recently led to the design of apprenticeship systems (Newman, 1989) and collaborative learning systems (Dillenbourg and Self, in press). The importance of environmental resources has been advocated by researchers within the 'situated' stream (Brown, 1990). The cognitive architecture of MEMOLAB gives a central role to patterns of interaction between the learner and the learning environment. The induction by the learner of interaction patterns is guided by external representations (Dillenbourg and Mendelsohn, in press).

2. Project Overview

Our research goals concern the design and implementation of a learning environment for acquiring basic skills in observation methodology for human sciences. We have chosen to develop an artificial laboratory, called MEMOLAB, in which learners can build experiments on human memory and run a simulation of these experiments. This learning environment aims to teach the methodology of experimental psychology. Basically, the learner's activities form a cycle: design an experiment, run it and analyze the output data.

MEMOLAB is based on a conceptual framework (see section 3. 1) which defines a content-independent cognitive architecture. During the first year, we devoted most of our resources to elaborating such a generalizable framework. During the second and third year of this project, the main focus is the implementation of the various components of MEMOLAB. We list below the main components of MEMOLAB.

- The LAB is a graphical interface where learners build an experiment by assembling objects on a workbench. The cognitive architecture is embodied in the lab by the type of objects that users manipulate to design an experiment. By changing the type of objects available, we have discriminated four LEVELS for the lab.
- The SIMULATION receives as input the experiment designed by the learner and returns the results of this experiment. The results are generated by comparing the learner's experiment to a knowledge base of experiments collected from the literature on human memory. This process is implemented by using case-based reasoning techniques (see section 3. 3).
- When results have been computed, the learner can analyse them by using tools: DATA-TOOL will provide basic data visualisation and data transformation facilities and STATISTICAL-TOOL will enable basic statistical processing.

During this cycle (design - simulate - analyze results), learners have access to theories on human memory and to a library of experiments that have been published. This encyclopaedia of MEMORY is implemented in hypertext form. Users can find information at the level of detail they want.

In educational computing, this set of five modules is named a 'micro-world', i. e. a simplified world that learners can explore without any tutorial control. Experience has shown however that, in order to guarantee pedagogical efficiency, two conditions must be satisfied. First, the micro-world must be structured. In MEMOLAB, the structure is formed by the LAB levels, reflecting the system's cognitive architecture. The second condition is that learners receive guidance when they need it. In MEMOLAB, we adopted an original approach to this issue by distributing guidance over a set of agents. The difference between agents and tools is that the former can take the initiative while the latter are only used on learner request.

- The main COACH has to set up an environment which provides optimal learning conditions for a given learner, at a given time. He cares for instance about the learner progression within the sequence of lab levels.
- In addition, MEMOLAB includes three specific tutors, which will provide assistance on particular aspects: 'MASTER of DESIGN' will critique the experiments designed by the learner, 'STAT-GIRL' and 'DATA-KING' will respectively support him during the data analysis and statistical processing stages.

Managing the interaction between these agents represents a serious problem. A BLACKBOARD component will control this society, solving priority conflicts and circulating data among agents (see section 3. 6).

3. Completed Work

3.1. The Conceptual Framework

Designing an intelligent learning environment (ILE) involves implementing some theory of learning and teaching. However, most available theories do not have the level of operationality required for implementation work.This is why the design of an ILE remains a real research work. Our challenge is to develop a framework that builds a bridge between theories and implementation. This framework translates psychological knowledge into terminology more relevant to the computers scientist. It specifies the cognitive architecture of MEMOLAB.To present a detailed view of our intermediate framework goes beyond the scope of this report. A detailed analysis can be found in the publications listed in section 4. The framework is built around two related concepts: the *pyramid metaphor* and the *language shift* mechanism.

The pyramid is a visual metaphor for the core structure of a learning environment. The pyramid represents the concepts and skills to be acquired by the learner, ranked bottom-up according to their level of "hierarchical integration" (see below). Learning consists in moving up in the pyramid. Each level of the pyramid is defined by two languages: the *command language* and the *description language*. The command language vocabulary is the set of elementary actions that the learner is allowed to do at some stage of interaction. The command language syntax defines how the learner composes sequences of elementary actions. The description language is the set of symbols (strings, graphics,...) used by the computer to show to the learner some description of her behavior. This description reifies some abstract features of the learner's behavior in order to make them explicitly available for metacognitive activities (Collins and Brown, 1988).

The command and description languages are different at each level of the pyramid. The hierarchical nature of a pyramid implies that each level integrates its lower neighbor. This integration is encompassed in the relationship between the languages used at successive levels: if a description language at level L is used as a new command language at level L+1, then the learner is compelled to use explicitly the concepts that have been reified at level L.This is what we called the language shift mechanism (Dillenbourg, to appear): when she receives a new command language, the learner must explicitly use the concepts that were implicit in her behavior. The meaning of the new commands has been induced at the previous level by associating the learner's behavior with some representation. This representation is now the new command. The ILE structure can then be described as a sequence of [*action language, representation language*] pairs, a sequence in which the relationship between two successive pairs is described by the language shift mechanism.

Let us consider a simple example on solving equations. At some level of the pyramid, one can show the learner - with some graphics - that a good heuristic is to regroup the X's on the same side of the equation. At the next higher level, we can offer a button "regroup X" in order to compel her to explicitly use this heuristic in her solution. A more complete example will be described when presenting MEMOLAB.

The process by which properties that are implicit at some level of knowledge can be abstracted and explicitly accessed at the higher level has been studied under the label of *reflected abstraction* (Piaget, 1971). The language shift mechanism has two uses. Firstly, it translates this psychological concept in a terminology more relevant for ILE designers. Secondly, it describes a pedagogical strategy (mainly inductive) to trigger reflected abstraction.

By applying the framework to ILE design, we ground the structure of this learning environment in a model of cognitive development. Reciprocally, this model of development can be tested through the difficult process of implementation. We found that this intermediate framework can be used as the interface between several theoretical backgrounds. Most psychological theories address actually only a specific facet of learning while an ILE designer must consider learning in its globality and complexity. Therefore, an intermediate framework should integrate multiple theoretical bodies of knowledge, each relevant for some aspect of reality. For instance, an ECS must account for the importance of discovery, for the role of practice and for the effect of coaching, because all of them occur at some stage of learning in the real world. The framework we propose can be read from different theoretical perspectives. We outline two of these psychological theories and concentrate on the one that is central for MEMOLAB: the theory of Robbie Case (1985). In Dillenbourg (to appear), we also relate our framework to research in AI, especially to learning from solving path systems.

From Campbell and Bickhard's (1986) viewpoint, the language shift mechanism can be viewed as a process of inducing *interaction patterns.* An elementary interaction associates some sequence of user's actions and the computer's description of this sequence. We showed that inferring the meaning of the description language can indeed be described as the result of inducing the relationship between the actions performed and their representation (Dillenbourg, to appear). This corresponds to a view of knowledge as something that stands in the interaction between the subject and her environment. It creates a bridge between our model and current research on *situated learning* (Brown,1990), a "hot" issue in AI and Education (especially on the West Coast).

Our intermediate framework also introduces the designer to the theories of Vygostsky. The *apprenticeship* idea is reified in the pyramid model by sharing control between the coach and the learner: when the learner is able to perform at some level L, the tutor must guide her activities at level L+1. This level L+1 corresponds to the concept of zone of proximal development (Vigotstky, 1978). At each language shift, the learner will assume a more important control of his solution process and the coach's guidance will be reduced. Moreover, Wertsch (1985) proposed a linguistic analysis of the internalization process that relates it to the language shift. He observed (in mother-child interactions) that the move from the inter-individual to the intra-individual plane was preceded by a language shift inside the inter-individual level: mothers replace a descriptive language by a strategy-oriented language (i.e. a language that refers to objects according to their role in the problem solving strategy).

The third but central theoretical background that fits with our framework is the neo-Piagetian theory of Robbie Case (1985). We focussed on this theory because it has already a rather operational form. The key idea in Case's theory of intellectual activity and development is what he calls the "executive control structure". He believes that there is a valid general analysis of problem solving across domains viewed as the execution of a "mental plan". This plan is defined as a sequence, or program, of schemata. There are two types of schemata: "figurative schemata" represent states and "operative schemata" represent transformations. The mental plan is divided into three main sub-components.

- A representation of the "problem situation": this is the set of conditions relevant to the plan. The complexity of the representation will depend directly on the complexity of the problem.
- The goals to be achieved defined as a set of new states, or "solution situation".
- The "problem solving process" to be used, stated as a set of operations that transform the problem situation into the solution situation.

These components are further analyzed. Elements of the problem situation are mapped to elements in the solution situation, and both are mapped to transformations in the strategy set. The result is a well-defined formal structure that allows Case to associate specific tasks with problem solving processes in a rigorous way.

Case formulates his general theory with reference to developmental stages as they are identified in specific domains. One of the characteristics of his theory is that it relates ***quantitative changes*** within a stage to ***qualitative changes*** between stages: for example, an increase in the active unit capacity of working memory occurs within a stage, but helps to explain the transition to the next stage. Case distinguishes activity within a stage (i.e. a "sub-stage") by first defining what he calls "basic units of thought". He then notes that during development (and probably also during skill acquisition) we have the classical four stages:

- Perception of objects and motor activities
- Relations between motor activities
- Manipulation of dimensions (quantifiable variables)
- Second order dimensions (ratios)

How do we explain the formation of new units and the *transition between stages*? According to Case, each new sub-stage within a stage is characterized by the subordination of a new basic unit to the executive control structure: the first sub-stage has two basic units, the second has three and the third has four. The complexity of subordination reached at the final sub-stage (in stage n) is such that it corresponds to a basic unit at the next stage (stage n+1). When the executive control structure of stage n+1 subordinates two of these basic units passed up from below, it will enter its own first sub-stage... and so on. The last sub-stage of stage n can thus be considered as sub-stage zero of stage n+1. In other words, the four-unit control structure of stage n can be translated into a one-unit control structure at stage n+1. It is this formal process which Case calls “hierarchical integration”.

It is an increase in “Short Term Storage Space” (STSS) that permits the transition from one sub-stage to the next. This increase is achieved within the “Total Processing Space” (TPS) which also contains the “Operating Space” (OS) utilized to control the active schema. STSS increases with age during development as a result of the maturation of the nervous system. It also increases during the learning of schemata as the result of an increase in the efficiency of the control structure: as the learner masters a task, the compilation of her knowledge frees up *short term memory* to hold new objectives.

There is an obvious mapping between the structure defined by Case and our intermediate framework.The control structures at each level of the pyramid integrate the control structures located at the lower level. The sequence of microworlds within the pyramid is structured as Case’s view of development: quantitative variations define the improvement possible within some level (or microworld or stage) while the qualitative variations define the transition between two levels.The concept of stage transition is translated into the language shift mechanism. This transition is necessary when the learner try to solve problems that have too high memory load constraints. After the language shift, the learner has at her disposal new control structures that enable her to solve the problems with a reduced cognitive load. This ILE-oriented re-expression of Case’s allow us to ground the design of MEMOLAB in this theory.

3.2. The MEMOLAB Design

The pedagogical goals of MEMOLAB are that students acquire the fundamental concepts necessary to perform experimentation in psychology. These concepts are experienced by designing and running experiments on human memory, in an *artificial lab*. Globally, the learner’s activity is cyclic. She starts with a challenge that she specifies or that is proposed by the coach. Then, she selects some subjects, the list of words to be presented, the subject activities, how performance will be measured, etc.

Here is an example of an experiment that the learner can design:

> Present a list of 10 words to two samples of 30 subjects. For the experimental group, the words are selected within the same semantic field. For the control groups, words have no close semantic relation. Let the subjects study the list during 5 minutes and will have to identify the first list words in another list of thirty words. Performance will be measured as the number of words correctly identified.

Then the system will run the experiment. This may be done in two non-exclusive ways: the system simulates the behavior of subjects, or it presents the experiment to the learner which plays the role of a real subject (but is aware of the experimental goals). The last stage of the learner’s activity cycle is to analyze the collected data in order to reach some (temporary) conclusions with respect to her hypothesis.

We have structured this learning environment according to the principles of our framework. The MEMOLAB pyramid has three levels, plus a level-zero devoted to learning how to use the system itself. This pyramid is built around the concepts of experimentation. Creating an experiment means assembling concrete events on the lab workbench. An event is the association of some subjects with some task and a measurement device. A task is defined by an activity, a material and some parameters. Building a plan (level 3) means creating a non-instantiated sequence on the workbench and generating the experiment by defining a n-dimensional table of factors (n being the number of independent variables). Peripheral concepts from statistics and human memory are attached around these core concepts. The three levels are:

- 1: Building non-comparative experiments (single sample)
  Units: events
- 2: Building comparative experiments (one factor, several samples)
  Units: sequences or paradigms
- 3: Building bi-dimensional experimental plans
  Units: plans

An overview of the learner's activities at each level of the pyramid is presented in figure 1. For each level, we schematize two screens, one displaying the command language and the other presenting the description language.

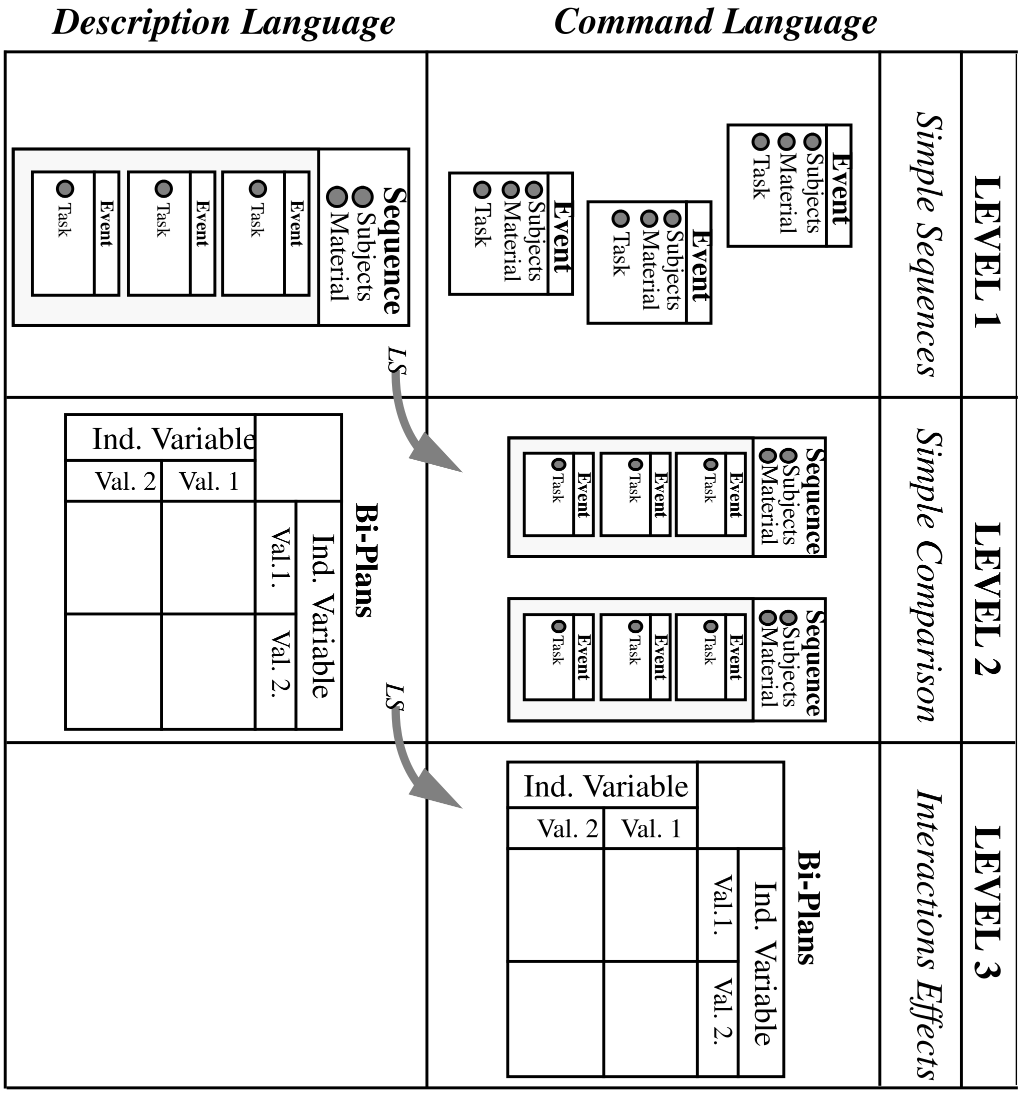

Figure 1 :The Cognitive Architecture of MEMOLAB

At level one, the novice sequences event-frames on the lab workbench. Each event-frame includes some part of the information that defines the event (subjects, task,...). At the end of level 1,the learner receives challenges that induce the need for comparisons.The concept of sequence is reified at level one and used at level two as the way of designing new experiments. At the end of the second level, the necessity of taking into account the interaction of effects is induced by new challenges.The concept of plan is presented at level two and used for designing experiments at level 3.

The shift from one level to another, i.e. to shift from one language to another corresponds to some *qualitative* jump in learning. Within each level, we defined four sub-levels that are discriminated by *quantitative* differences. These differences result form an increase in the difficulty of the challenges proposed by the coach. More complex challenges compel the learner to handle a larger number of dimensions and hence increase the working memory load. At the end of the second sub-level, the learner receives challenges that already belong to the next level. This shows the learner the necessity to have more powerful control structures to solve the proposed challenge (As in Case theory sub-level i.4 is equivalent to sub-level i+1.0).The "reunitarisation" of the objects used at some level in a new more powerful object frees the memory resources necessary to solve the problem.

We have tested many layouts for the interface, the first ones have been drawn manually and the later ones have been programmed. The design of the main lab window is a compromise between the cockpit model and a Macintosh-like interface. In the cockpit model, learners permanently have all information available on the screen. At the opposite, on Macinstosh-like interfaces, one has to pull-down a menu in order to find the options available. The former reduces the mnemonic load during problem solving, but increases -until saturation- the complexity of the screen. The latter keeps the screen clear, but implies the user must know what exists under each menu. Menu operations also take more time than direct object manipulations. We choose the cockpit model for the objects that are directly involved in the experiment design: groups, tasks, materials, events, sequences and plans. The operations which are less often performed, like a creating a new experiment, are placed in a menu.

3.3. Hypertext

A hypertext system was deemed the most appropriate computational infrastructure for the 'Encyclopaedia of Memory', the 'Handbook of Experiment Design', and other on-line information tools. In traditional text processing systems, a document is no more than a sequence of character strings which are stored and retrieved in a strictly linear fashion. The essential innovation of hypertext consists in violating this linearity constraint. The hypertext approach views a document as a set of text chunks corresponding to conceptual nodes in a database; a system of machine-supported links between nodes and chunks allows interactive branching within and between documents. Hypertext systems exploit the latest developments in computer interfacing technology, particularly window management and graphics tools. Conceptual entities are displayed as windows of text which may contain any number of icons or buttons representing links to other nodes. The user can navigate freely around the hyperdocument by clicking on any of these graphics pointers.

While commercial hypertext shells are available, we have implemented our own hypertext system for several reasons. First, building the encyclopaedia on a commercial shell would lead to portability and copyright problems when we distribute copies of MEMOLAB for testing and validation purposes. Second, we think an intelligent learning environment should give an overall impression of implementational coherence; by building our own hypertext tool, we can freely tailor it to share the look-and-feel of the other system components, especially the LAB interface. Third, commercially available systems are usually closed products, and embedding them in a larger system leads to inevitable communication problems. This is a crucial point in MEMOLAB, where the encyclopaedia should interact effectively with the other modules. For example, when the tutor feels that the student needs additional information on a given subject, she should be able to select and display the relevant sections of the encyclopaedia. Conversely, the hypertext component should be able to call on other agents to provide intelligent tutoring functions. For instance, if the currently displayed text concerns a well-known experiment, the student can click on a 'RUN' button to simulate it; in this case, the hypertext system should be able to call on the LAB and the SIMULATION to fulfil the student's request.

In MEMOLAB, the entire hypertext subsystem is implemented as a class of objects called hypernodes. Each hypernode contains a title and a chunk of text with embedded formatting commands. Moreover, each node has an associated list of keywords summarizing the semantic content of its main text. These keywords serve as the basis of an indexing facility that not only allows the student to browse around the encyclopaedia, but also enables the tutor to determine complementary readings to recommend, depending on the current student diagnosis.

To optimize the performance of the hypertext module, the text of each hypernode is preprocessed by a special-purpose parser which transforms the source text into an executable LAMBDA function. Thus a single function call replaces time-consuming on-line formatting whenever a node has to be displayed. As demonstrated by run-time statistics gathered at different stages of the implementation process, this once-and-for-all parsing technique reduces display time by a factor of 7.

While fully integrated with MEMOLAB, the hypertext module can be used on a stand-alone basis for other applications. It may thus be considered as one of the spin-off products of the MEMOLAB system that may prove useful independently of the current project.

3.4. Case Based Simulation

The input of the simulation is the experiment that the learner has built in the LAB. In an experiment, pseudo-subjects are supposed to memorize words or other items and to try later to remember them. The output of the simulation is the list of words that each subject has remembered. The component that performs the simulation is called the Memory Machine (MM). The constraints that justify our design choices are related to the pedagogical function of MM and to the nature of available knowledge.

- ***Pedagogical constraints.***

The simulation has to produce results as similar as possible to the results that one would obtain if the experiment was performed on real subjects. However, psychological experiments have *de natura* a low fidelity. Moreover, this simulation is clearly not a substitution for real experimentation. Therefore, the validity of simulated results must be assessed in the light of pedagogical goals: to acquire basic skills in methodology of experimentation. This goal determines the lower limit for the validity of MM results. This lower limit can be expressed by the following rule:

> Minimal Validity Rule: if the learner builds an experimental plan with a factor F and based on a paradigm P, and if the literature includes knowledge on the effects of factor F, within the paradigm P, then the simulation must produce data in which the effects F can be identified.

More than being precise, the results must be explainable. Several ILE designers (White and Frederiksen, 1988; Roschelle, 1988) have shown that a qualitative simulation, although less efficient, is better suited to pedagogical purposes. Simulating an experiment involves an analysis of the structure and content of this experiment. The simulation produces a trace that can be used in explanation or in diagnosis of the learner's experiment mistakes. This trace consists of prestored comments and keywords. 'Master-of-Design' will parse the trace on the basis of these keywords. We are exploring the possibility for MM to generate a trace structured in hypertext form. This would allow the learner to read the explanation at the level of detail that she wants.

- ***Knowledge Acquisition***

Contrary to many physical devices, human mnemonic behaviour cannot be predicted by a set of formulas or rules. Available theories do not constitute a consistent and exhaustive body of knowledge. Knowledge is distributed among a large set of experiments. Therefore, learners' experiments will be simulated by comparison to a similar experiment from the literature. This process is implemented with cased-based reasoning (CBR) techniques (Riesbeck and Schank, 1989).

Moreover, the literature does not cover the very large space of experiments that can be designed in MEMOLAB. Within the space of possible experiments, psychologists have concentrated their work on some avenues, or paradigms. (In experimental psychology, a paradigm is defined as a class of experiments.)The comparison of experiments within a paradigm brings more information than comparing experiments which differ in too many ways. Therefore, the case-retrieval process is not a simple search through a flat set of cases, with some similarity metrics. The set of cases (literature experiments) is partitioned into paradigms and a main stage during the simulation consists in identifying the relevant paradigm (see below, the paradigmatic analysis).

Another peculiarity of this domain is that the adaptation of the retrieved case to the target case cannot be driven by universal rules. Let us imagine that the learner builds an experiment where subjects have to memorize a long list of words and that MM retrieves an experiment which is similar on all points but the list length. The effect of list length is not universal, but varies according to other factors such as the delay between the memorizing and recall events. Therefore, case-adaptation is performed by a set of rules (named Vertical Adapters) that differ for each paradigm. In short, case-adaptation in MM is governed by rules that are themselves case-dependent.

- ***The Case Library***

The library includes the paradigm discrimination tree (figure 2) and a set of experimental sequences (cases). Each leaf of the paradigm tree corresponds to a subset of sequences found in the literature. When we have identified to which leaf node corresponds the learner's sequence, we retrieve the experiment which is the most similar to that of the learner. The vertical adapters (VA) are stored at different levels of the tree, according to their generality. Indeed a set of VA forms a simple production system.

- ***The Algorithm***

The implementation revealed hard issues and led us to several designs. We implemented two of them. We describe the current implementation. MM must reason at the same time about the content and the structure of experiments. The content of the experiment refers to the sequences composing the experiment: what are the tasks, the material features, etc. This information is necessary to determine the paradigm to which the learner's experiment belongs. The structure of an experiment refers to the relationship between sequences. The fact that, for instance, two experimental sequences are identical on all points but the material length indicates that the learner intends to observe the effect of the material length. This 'factor' must be identified in order to respect the 'minimal validity rule' (see above).

The *structure analysis* returns the factors identified within the learner's plan and the value (modality)of this factor in each sequence. This knowledge feeds the *paradigmatic analysis* process which identifies the most specific paradigm (leaf node) corresponding to the learner's sequence. (The word ‘paradigmatic’ refers here to the meaning used in experimental psychology, as specified above.)

Let us consider first the simplest simulation process, i. e. computing the results for a sequence s_i. This sequence s_i is matched against the discrimination rule of each node N of the paradigm tree. If the match is positive, we store the vertical adapters of N and recurse on descending nodes. When a leaf node is reached, the associated set of literature sequences $S'_i = \{s'_{i.\,1}, s'_{i.\,2}, \ldots, s'_{i.\,n}\}$ is retrieved. In addition, vertical adaptation rules have been collected through the tree: $VA_i = \{va_{i.\,1}, va_{i.\,2}, \ldots, va_{i.\,m}\}$.

.

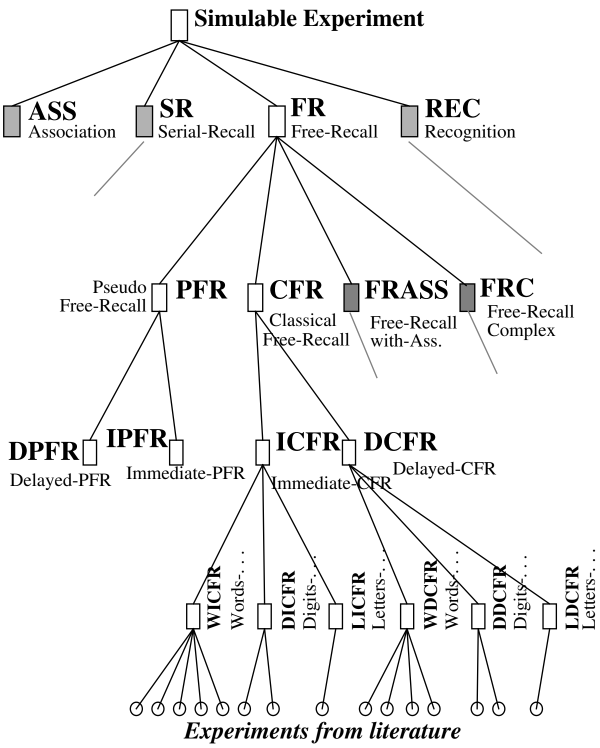

Figure 2 : The Paradigm Tree (as developed so far)

The selection process reasons on VA_i in order to select among S'_i the experimental sequence s'_i which is the most similar to s_i. The last stage is to retrieve the data of s'_i - denoted D'_i - stored with s'_i, and to execute VA in order to compute D_i:

$$VA\ (D'_i) \Longrightarrow D_i$$

Let us imagine that the learner’s experiment includes two sequences s_1 and s_2. We can apply independently the same algorithm on each sequence and generate results for each sequence:

$$VA\ (D'_1) \Longrightarrow D_1$$
$$VA\ (D'_2) \Longrightarrow D_2$$

However, this independent processing does not respect the minimal validity rule expressed above. Let us assume that between s_1 and s_2, the learner has created some systematic differences in order to observe the effect e of factor f (denoted e_f). With independent sequence processing, we cannot guarantee that, when learners will compare D_1 and D_2, they will find e_f. In order to guarantee that differences between D_1 and D_2 correspond to the expected effect, one must generate D_2 from D_1 and what we know about e_f. When s_1 and s_2 are compared to a tree node, the effects stored with that node are compared to the structure of (s_1,s_2). This structure was determined during the structure analysis stage. If e_f is identified, the Horizontal Adapters *(HAs)* that are associated with e_f are stored. Then, the experiment is 'reduced', i. e. only s_1 is passed to lower nodes. Data for s_2 will be generated from D_1 and the VA of e_f.

$$VA(D'_1) \implies D_1$$
$$HA(D_1) \implies D_2$$

These two strategies are summarized in figure 3. The optimal strategy is applied when a known effect is identified within the learner's experiment and if this effect does not interact with other factors. The optimal solution can be applied on parts of the learner's experiments.

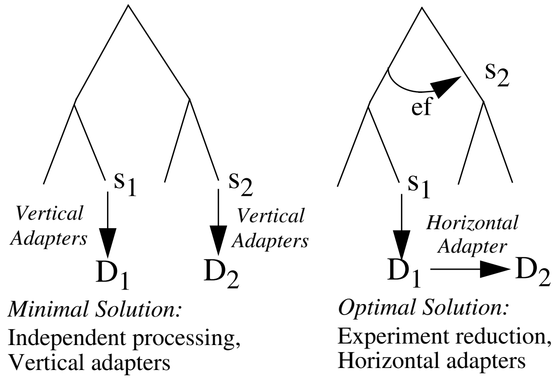

Figure 3 : Strategies for the Paradigmatic Analysis

3.5. Coaching and Tutoring

Designing an intelligent learning environment is attempting to solve a paradox. On one hand, we know that knowledge cannot be simply transmitted but has to be reconstructed by the learner, through experience and reflection. But, on the other hand, ten years of LOGO practice has shown that the work done by the 'animator' determines the learning outputs. The role of the animator is to set up a environment which will provide optimal learning conditions for a given student, at a given time. Moreover, at some stages, he has to give an advice on a particular point. We have distributed these roles among several agents: one coach and three tutors. The coach is a manager of learning while tutors are local advisers.

The coach is responsible for maintaining the whole learning environment in a state that offers optimal learning conditions for a given learner. The coach does not directly interact with the learner. He does not 'teach' or 'explain'. He plays an indirect game: modifying the environment in order to influence the learner's activities. The efficiency of decisions taken by the coach depends upon the careful design of the environment. This justifies the attention we paid to the cognitive architecture of MEMOLAB during the first year of research. The coach cares for instance about the learner's progression within the sequence of lab levels. He decides the moves from one level to another, i. e. the 'language shift', on the basis of the learner's work. He also decides which 'challenges' can be proposed to the learner. Experience has shown that most learners need to be invited to solve harder problems than those they are already able to solve.

Contrary to the coach, tutors will be more directive, i. e. they will tell the learner what is wrong. They are content-dependent, each of them provides advice in its expertise area. The 'Master-of-Design' will critique the learner's experimental design, 'Data-King' is a specialist in data manipulation and ‘Stat-Girl’ helps the learner in statistical processing.

The agents form a hierarchy in which the coach has more authority than the tutors. The coach can for instance decide to inhibit ‘Master-of-Design’ telling the learner what’s wrong in his plan, in order to leave the learner to search by herself for the reason why the plan failed.

This distribution of external guidance among several agents fulfills several criteria:

- As specified in our original research project, we aim for an intermediate solution between highly directive tutoring systems and free discovery learning environments. This task distribution implements this intermediate approach. It recognizes the learner’s activities as the source of learning without discarding the need for external advice.
- Our original research project also mentioned our aim to produce a generalizable system. The discrimination between a general, content-independent coach and several specialized tutors, offers a modular solution for generalizing MEMOLAB: MEMOLAB’s core structure will be reusable by joining new specialized tutors to the main one.
- Researchers on Intelligent Tutoring Systems have shown that implementing an intermediate level of guidance was far more difficult than implementing extreme approaches (free discovery or directive teaching). By distributing roles among agent, we guarantee that each agent will have some internal consistency, some personal teaching style, and thereby will be easier to implement.

3.6. Blackboard and Inference Engine

As seen in the preceding sections, MEMOLAB is a complex system whose overall behaviour emerges from the interaction of several specialized agents. This calls for a control structure that meets the following requirements: first, it should allow for opportunistic decision-making based on the coach’s assessment of student progress as well as local constraints provided by the individual agents; second, it should provide a uniform mechanism for integrating diverse reasoning methods and strategies. We are convinced a blackboard architecture provides precisely the control mechanisms we need to coordinate a variety of knowledge sources, each with its own data structures and processing tools. With this in view, we have tried several existing blackboard shells but have found them inadequate. More specifically, GBB and BB1 are implemented in pre-CLOS Common Lisp, and using them would pose a major compatibility problem, since the rest of MEMOLAB is implemented in CLOS. We would either have to relinquish the advantages of object-oriented programming or adopt a duplicate representation: each agent’s knowledge would be represented in the form of objects for local processing needs, but would have to be translated into the more conventional list-oriented structures of these blackboard shells for purposes of control and communication. The first solution was eliminated because CLIM, the interface manager used to implement the lab and the hypertext module, is based on CLOS; the second solution was judged too inefficient to be retained. Moreover, using a commercial blackboard would have raised royalties problem for distributing MEMOLAB.

We thus decided to implement our own object-oriented blackboard system.

In designing our blackboard control mechanism, we had to foresee more than the minimal facilities usually furnished with commercial blackboard shells. In general, these are empty control shells which ensure coordination of diverse knowledge sources by specifying a standard communication format, that of the messages to be posted on the blackboard(s). However, the choice and implementation of the individual knowledge sources’ computational tools are left to the user. In the case of MEMOLAB, its knowledge sources draw from a wide variety of reasoning techniques: case-based reasoning for the simulator, procedural interface management for the lab, rule-based inference for the main coach, whereas the specialized tutors shift between rule-based and algorithmic reasoning according to the nature of the specific problem at hand. Each of these processing tools had to be implemented, but special attention had to be given to rule-based reasoning, which is called upon by several knowledge sources. Hence the decision to merge rule-based reasoning and blackboard control in our implementation agenda.

As a first step towards this blackboard-controlled production system, we have implemented an inference engine with the following main features:

- ***tight integration of object-oriented and rule-based programming***

In first-generation rule-based systems, the inference engine was assigned a special working memory, neatly separated from other data structures such as those manipulated by conventional code. This working memory was represented as a list of assertions which were matched with rule conditions to determine applicable rules. The advent of object-oriented languages led to an intermediate version where a global memory of object structures coexisted with a working memory of logical assertions: only the latter was accessible to the inference engine. MEMOLAB strives towards full integration of object-oriented and rule-based representation. On the one hand, rules should refer to and work directly on the hierarchy of classes and objects present in the global memory. On the other hand, rules themselves, their antecedents and their conclusions, as well as the other task-implementation structures needed to operate the inference engine (ex. the current conflict set) are all represented as objects. As a result, this integrated object-oriented, rule-based system provides:

- ***full computational support of metareasoning***

Since rules are represented in exactly the same formalism and within the same global memory as domain objects, rules can reason about rules and can examine their own (and other rules') components, evaluation contexts, history of past successes and failures, etc. Rules can also modify (even destroy) themselves and other rules as well as create new rules. Full metareasoning power is not available in classical production system languages: an OPS5 rule, for example, cannot directly examine its own instantiations because these are represented differently and separately from domain knowledge. To implement metalevel reasoning in such systems, programmers usually have to code special-purpose functions to retrieve metaknowledge inaccessible to rules.

- ***a powerful rule representation language***

To give an idea of the expressive power of the representation language used, the basic rule condition template is the following:

(<object> <class> <slot$_1$> <operator$_1$> <value$_1$>
. . .
<slot$_n$> <operator$_n$> <value$_n$>)

where <operator> can be =, <>, <, >, <=, or >=.

This rule condition is the object-oriented equivalent of a set of two-place logical assertions where <slot> is the predicate, <object> the first argument and <value> the second argument. Note that here, the rule interpreter allows variables not only for the arguments but also for the predicate, thus attaining the power of second-order predicate logic. In addition, a rule condition may include any function call:

(CALL <function> <argument>* <operator> <result>)

Universally and existentially quantified rule conditions are also allowed:

(FORALL (<variable>*) (<var-domain-condition>*)
<subcondition>*)
(FORSOME (<variable>*) (<var-domain-condition>*)
<subcondition>*)

Finally, any of the above rule conditions (or any conjunction of these) may be negated.

The most important rule actions consist in creating or destroying objects, or modifying slot values of objects. In the general case, these actions are applied to objects of the application domain; they are then base-level actions. But since rules themselves are represented as full-fledged objects, these same actions become meta-actions when applied to rules; that is, the same set of commands allow us to change rule conditions and actions, rule priorities, etc. Furthermore, metalevel-specific actions, like activating a particular rule or rule set, allow the user to modify the inference engine's control strategy. Finally, to enhance the system's flexibility, any user-defined function can be called from a rule's right-hand side.

- ***possibility of concurrent multi-expert reasoning***

Up to this point, we have referred to 'the' inference engine as though only one were possible in a given system. Actually MEMOLAB's production system framework has been designed with a view to implementing multi-expert systems. Any number of inference engines can be created, each associated with its own knowledge base (KB). For example, we could generate two instances of the class of inference engines to model two different tutors working concurrently on the same problem, each with her own knowledge base and problem-solving history. The inference engine specially dedicated to overall control will be a particular instance of the class of inference engines. The blackboard system can be managed by such a control-KB/ meta-engine tandem, while the domain knowledge sources can run their individual knowledge bases on (an)other inference engine(s).

With the above features, MEMOLAB's inference engine attains a level equal to that of the more advanced expert system shells on the market - at least as far as knowledge representation and reasoning strategies are concerned. It may prove to be another non-trivial spin-off from the MEMOLAB project

4. References

BENTO C. et COSTA E. (1988) *Automatic Cognitive Modelling in Hierarchical Domains*. Proceedings of ECAI'88.

BLOCH G. and FARRELL R. (1988) *Promoting Creativity through Argumentation*. Proceedings of ITS-88, Montreal,pp. 243-249.

BRAZDIL P. B. (in press) Integration of knowledge in multi-agent environments. In E. Costa (Ed.). *New Directions for Intelligent Tutoring Systems*. Berlin: Springer-Verlag.

BROWN J. S. (1990) Toward a new epistemology for learning. in C. Frasson and G. Gauthier (Eds). *Intelligent tutoring systems at the Crossroad of AI and Education*. Norwood, NJ; Ablex.

CAMPBELL and BICKHARD (1986) *Knowing Levels and Developmental Stages*.Karger.Basel

CASE R. (1985) Intellectual Development form Birth to Adulthood. Academic Press. New York.

CLANCEY (1983) *The advantages of abstract control knowledge in Expert system design*. Report Nº STAN-CS-83-995, Stanford.

COLLINS A and BROWN J. S. (1988) The Computer as a Tool for Learning through Reflection, in H. Mandl and A. Lesgold (Eds), *Learning Issues for Intelligent Tutoring Systems. Springer* Verlag. New York, pp. 1-18.

CONKLIN, J. (1987). Hypertext: An Introduction and Survey. *Computer*, Vol. 20, No. 9, pp. 17-41.

COSTA E. and URBANO P (in press) Machine learning, explanation-based learning and intelligent tutoring systems. in E. Costa (Ed) *New Directions for Intelligent Tutoring Systems*. Berlin: Springer-Verlag.

DILLENBOURG P. (1989) Designing a self-improving tutor: PROTO-TEG. *Instructional Science*, 18, 193-216.

DILLENBOURG P. (in press) The Language Shift: a mechanism for augmenting learner's control and abstraction. To appear in P. WINNE and M. JONES (Eds). *Foundations and frontiers in Educational Computing Systems*. Berlin: Springer-Verlag.

DILLENBOURG P. and MENDELSOHN P. (in press). The genetic Structure of the Interaction Space. To appear in E. Costa (Ed). *New Directions for Intelligent Tutoring Systems*. Berlin: Springer-Verlag.

DILLENBOURG P. and SELF J. A. (in press) Designing human-computer collaborative learning. in C. E. O'Malley (Ed), *Computer-Supported Collaborative Learning*. Wiley & Sons.

HARNAD S. (1990) The symbol grounding problem. *Physica D*, Vol. 42, N 1-3, pp 335 - 346.

HRON A. (1990) *A diagnostic system for qualitative knowledge*. Paper presented at the third ITS Seminar in Arhus, Denmark, 5 - 9 December.

MENGEL S. and LIVELY W. (1991) On the Use of Neural Networks in IntelligentTutoring Systems. *Journal of Artificial Intelligencein Education*. Vol. 2. , N 2. , pp. 43 - 56.

NEWMAN D. (1989) *Is a student model necessary ? Apprenticeship as a model for ITS.* Proceedings of the 4th AI & Education Conference (pp 177 - 184), Amsterdam, The Netherlands: IOS.

PIAGET J. (1971) Biology and Knowledge. The University of Chicago Press. Chicago

RIESBECK C. K. and SCHANK R. C. (1989) *Inside Case-Based Reasoning*. Lawrence Erlbaum Publishers. Hillsdale, New Jersey.

SELF J. A. (1991) *Formal approaches to learner modelling*. Technical Report AI-59, Computing Department, University of Lancaster (UK).

STEELS L. (1990) *Components of Expertise.* AI Magazine, Vol. 11, n 2, pp. 28 -49.

VYGOTSKY L.S. (1978) The Development of Higher Psychological Processes (edited by M.Cole, V. John-Steiner, S.Cribner and E. Souberman). Harvard University Press. Cambridge. Mass.WERTSCH (1985)Adult-Child Interaction as a Source of Self-regulation.in S.R. Yussen (Ed) *The Growth of Reflection in Children.* Academic Press. maison, Wisconsin

WERTSCH J. (1985) Adult-Child Interaction as a Source of Self-Regulation in Children. in R. Yussen (Ed). The growth of Reflection in Children. Academic Press, pp. 69-97.

5. Research Team

TECFA is an academic team active in the field of educational technology. It belongs to the School of Education and Psychology of the University of Geneva. It is directed by Professor Patrick Mendelsohn and includes about 10 collaborators. The team is in charge of the computer courses within the school and provides services within the computer-based learning community. TECFA main research interests are the applications of artificial intelligence to education, the cognitive effects of educational software and the communication issues with new technologies(distance education, multi-media systems,...).

This research project is funded by the Swiss Research Fund under a special research program (PNR 23), intitled ‘Artificial Intelligence and Robotics’. The contractors are Prof. P. Mendelsohn and D. Schneider. The research team is multidisciplinary. It includes P. Mendelsohn (psychology), B. Borcic (computer science), M. Hilario (computer science), D. Schneider (political and computer sciences), P. Dillenbourg (educational and computer sciences). The project will end in October 92.

Address for correspondance: TECFA, Faculte de Psychologie et des Sciences de l’Education, Universite de Geneve, 1211 Geneve 4. Switzerland. Phone +41.22.705.76.42. Fax: +41.22.20.29.27 E-mail: pdillen@divsun.unige.ch.