The Design of a Self-Improving Tutor : PROTO-TEG

By Pierre DILLENBOURG

Unité de Technologie de l'Education
Université de l'Etat à Mons (Belgium)

Abstract

This paper presents the principles and the architecture of PROTO-TEG, a self-improving tutor in geometry. This system is able to discover the criteria useful for selecting the didactic strategies it has at its disposal. These criteria are expressed as characteristics of the student model. They are elaborated by comparing student model states recorded when a strategy was effective and those recorded when the same strategy was not effective. This comparison is performed by machine learning methods, more precisely by learning concepts from examples. An empirical experiment was performed in order to assess the designed self-improving functions and conditions were discovered for five of the nine didactic strategies. However, this new knowledge did not lead to PROTO-TEG being more efficient in terms of student performance.

Published in Instructional Science, vol. 18, n°3, pp. 193-216.

This work was done in the Unit of Educational Technology, State University in Mons (Belgium), under the direction of Prof. L. D'Hainaut and Dr. C. Depover. The author's current address is : Faculté de Psychologie et des Sciences de l'Education, Université de Genève, 1211 Genève, Switzerland.

Some parts of this paper have been presented to the ITS 88 Conference, in Montreal, June 88, and other parts have been presented to the second European seminar on Intelligent Tutoring Systems, in Le Mans (France), October 88.

Introduction

A characteristic quality of a good teacher is her ability to improve her teaching activities as a result of her experience. Several authors emphasized that computerized tutors should be capable of self-improvement (Hartley and Sleeman, 1973 ; Self, 1977 ; Howe, 1972 ; O'Shea, 1982 ; Kimball, 1982 ; Stubbs and Piddock, 1985). Nevertheless, little research has been done on self-improving tutors since the work of O'Shea (1979) and Kimball (1982) done in the early seventies.

We report here our attempt to design a self-improving tutor in geometry called PROTO-TEG (this name comes from the French sentence "PROTO-type de Tutoriel Evolutif en Geométrie"). The first section of this paper locates PROTO-TEG and other self-improving systems in the context of knowledge acquisition within an Intelligent Tutoring System (ITS). The second section analyses our particular approach with respect to the issue of student modelling in an ITS. Then, we describe our system (section 3) and report the experiment we have carried out to evaluate PROTO-TEG's self-improvement (section 4).

1. Knowledge acquisition and self-improvement within ITSs

Self-improvement may be viewed as a particular result of knowledge acquisition (or learning). We will locate PROTO-TEG's self-improving process among the different learning processes performed by ITSs. For this purpose, we will use a three dimensional model. Its first dimension considers the content of learning and refers to the classical ITS structure. The second dimension describes the learning strategy in the terminology from machine learning research. The third dimension differentiates the various systems with respect to the generalizability of the acquired knowledge.

1.1. The first dimension : the content of learning.

As generally asserted, the knowledge in an ITS includes four sets of knowledge: the domain model, the tutoring model, the student model, and the interface managing the student-machine dialogue. This conceptual structure allows us to define four categories of learning processes with respect to the role of the knowledge acquired in the system (Duchastel and Imbeau, 1986; Dillenbourg, 1989).

Most systems are able to acquire knowledge about the student in the context of student modelling. Some systems are able to acquire knowledge for the domain model : the Self-Improving Tutor for Symbolic Integration (Kimball, 82) records the student's solutions if they are better than the solutions known by the system ; the GEO system (Duchastel and Imbeau, 1986), which teaches Canadian geography, acquires new information by asking the students questions about their own local geography. Another ITS, the Self-Improving Quadratic Tutor designed by O'Shea (1979), was able to acquire tutoring knowledge in order to improve the efficiency of its tutoring strategy.

With respect to this dimension, PROTO-TEG may be located between the student model and tutoring model categories, since the knowledge it attempts to acquire concerns the relationship between the student and the tutoring models.

1.2. Second dimension : the learning strategy.

The second dimension classifies systems according to their learning strategy, i.e. their way of acquiring knowledge. Among the various Machine Learning techniques available, the most used ones are the learning by instruction strategy (as in GEO and in the Self-Improving Tutor for Symbolic Integration, quoted in the previous section) and the learning by deduction strategy which has generally been applied to student modelling. More recently, several ITS designers have oriented their work towards inductive methods as we did for PROTO-TEG (e.g. Langley and Ohlsson, 1984). We may expect that this range of techniques applied to ITS will soon also include applications of learning by analogy and explanation-based learning techniques.

1.3. Third dimension : the "generalizability" of the acquired knowledge.

It is interesting to differentiate ITSs according to the extent to which the knowledge acquired by interacting with a student may be used with another student.

Let us imagine that a student working with GEO tells it that the Sainte-Rose-du-Nord village is on the riverside of the Saguenay. This information being true for one student, it will remain true for any other student who interacts with the system. The acquired knowledge may consequently be described as generalizable. On the other hand, when an ITS detects a bug in the student's knowledge, even if this acquired knowledge is valid for this student, the system cannot generalize its diagnosis to other students; they may have other bugs.

In other words, the generality level of acquired knowledge enables us to discriminate two categories of learning tutors :

- ***adaptive systems*** : if the acquired knowledge is not "generalizable", the function of the computer's learning is restricted to adapting itself to the student behaviour (and the tutor must revert to its initial state for each student) ;
- ***evolutionary*** or ***self-improving systems*** : if the acquired knowledge is generalizable, it enables a durable transformation of the system, which will improve in session after session.

PROTO-TEG belongs to the second category, but with some restrictions : the generality of the knowledge it acquires with a sample of students is actually restricted to the population from which the sample has been extracted. The above categories are not exclusive : the aim of PROTO-TEG is to improve its adaptive mechanisms.

2. A pragmatic approach to student modelling

PROTO-TEG had nine didactic strategies at its disposal, each implemented as the conclusions of a production rule. But initially, these production rules had no conditions part. The self-improving function aims to discover these conditions, i.e. to discover under what conditions (described by the student model) each strategy is efficient. This kind of learning corresponds to a "pragmatic" approach (Ohlsson, 1987) to student modelling.

From the didactic point of view, the role of student modelling is to help the teaching-learning model to make teaching decisions. As Self (1988) pointed out, *"the grand ambition to build high-fidelity student models can easily obscure the fact that, in practical terms, student models by themselves achieve nothing. Student models are merely data for the tutoring component of ITSs."*

The crucial task of an adaptive teaching system is to determine its next action according to the student's previous behaviour. The role of student modelling consists of organizing, structuring and summarizing the available information about the student. The result of this process is often a classification of the student's cognitive state with respect to a predetermined set (or "catalogue") of misconceptions, bugs, mal-rules, pre-representations, ... This condensed information simplifies the expression of laws for selecting didactic strategies, because the defined categories are meaningful for the designer. We must keep in mind that, for the computer, these categories are not meaningful but only useful.

Furthermore, building these categories requires a large amount of knowledge about :

- the set of possible bugs, misconceptions,... (student's cognitive states) ;
- how these states may be identified in the student's behaviour (diagnostic process) ;
- which didactic strategy is relevant for each cognitive state (interaction laws).

As ITS designers frequently discover, this knowledge is not directly available from cognitive psychology or educational science. Consequently they have to establish this knowledge by themselves. This process substantially increases the cost of building an ITS. Furthermore, this knowledge is generally collected by simple observation rather than by experimental methods, which increases the probability of incorporating errors in the system's knowledge base. This statement emphasizes the value of giving an ITS the ability to acquire a part of this knowledge by itself.

Our so-called pragmatic approach reverses the normal development of student models and the laws for selecting didactic strategies. Rather than starting from the student's behaviour and deducing the required strategies, we start from available strategies and try to discover for which set of behaviours each strategy is relevant. In this paper, this set of behaviours will be called a *student model category*. This approach requires the same pieces of knowledge as the one previously described, since the final aim remains to connect a strategy with a set of behaviours. Its interest lies in the fact that, in proceeding backwards, this knowledge may be partially acquired by the system.

This knowledge acquisition process is performed after the system has been used by several students. It is based on the system's ability to analyze the recorded interactions between the student and the computer and to determine for which set of behaviours a given strategy has been effective. The products of this learning task are the student model categories which constitute the criteria for selecting a strategy.

3. Presentation of PROTO-TEG

PROTO-TEG comprises two major components, the **tutoring system** and the **self-improving function**, implemented as separate programs. The student only interacts with the tutoring system, which records the student history. Later, the self-improving function explores the recorded files off-line with the purpose of extending the knowledge base of the tutoring part.

3.1 The tutoring part

PROTO-TEG goals are related to concept acquisition in geometry. The system is like a classical generative tutoring system, with an iterative structure. For each concept, the system chooses a didactic strategy, applies it and then verifies whether the student has acquired the concept. If he did, another concept is taught. If he did not, the system tries to teach the same concept again with another didactic strategy. The test used to verify concept acquisition consists of classifying successively 10 quadrilaterals as instances or non-instances of the concept.

Some aspects of the tutoring system have been simplified in order to focus our work on the self-improving function. Consequently, the tutoring part does not reach most of the standard requirements for an ITS label.

3.1.1 The domain model

The domain model includes ten concepts which are **classes of quadrilaterals**, each class being defined by one or two attributes. They have been arbitrarily created for this experiment in order to guarantee that the experimental and control groups (see section 5.) may be considered as equivalent with respect to their prior knowledge. The concept used in the experimentation are defined in table I.

Table I : Concepts taught in Proto-Teg

Name	Attributes
Brol	Two vertical sides
Malo	Two parallel sides and two isometric sides
Tchu	Two consecutive right angles
Ruca	One right angle and two isometric sides
Spec	Two vertical sides and two consecutive right angles
Buli	Two consecutive isometric sides
Arpo	Two right angles and two isometric sides
Goil	Two parallel sides and two consecutive isometric sides
Mika	Two parallel sides
Chol	One right angle and two consecutive isometric sides

This set of concepts were tested during a pilot experiment (see section 4). Two concepts (the "Goil" and the "Ruca") were withdrawn because they required a long study time and the concept set was divided into two sets in order to provide two learning sessions of about 50 minutes. The concepts were taught in the same order for each student :

Learning session 1 :	Brol	Malo	Tchu	Spec
Learning session 2 :	Buli	Arpo	Mika	Chol

The domain model also includes a problem generator, which builds quadrilaterals randomly and draws them at a specified screen location. The produced quadrilaterals possess the following characteristics :

- the positive instances may not have in common (by chance) any characteristics which are not included in the concept definition ;
- the length of the sides varies between 12 and 30 millimetres ;
- as far as possible, nearly-vertical and nearly-horizontal lines are avoided because they produce undesirable visual effects ;
- if an angle may not be a right angle, it will be greater than 97 degrees and smaller than 83 degrees in order to avoid student mistakes related to the imprecision of measurement ;
- for the same reason, the difference of length between two sides which may not be isometric will be greater than 5% of the sum of the length of each side ;
- the localization of the attributes on the instances must vary (for instance, the right angle may not always be the upper left one).

3.1.2 The tutoring model

As we have said previously, the starting point of the pragmatic approach is the following question : **which strategies may be designed to reach this goal ?** Since PROTO-TEG goals concern concept acquisition we have considered some results of cognitive psychology research in this area (e.g. Tennyson & Park, 1980)

These investigations and our personal teaching experience enabled us to select seven parameters which may differentiate two strategies : the number of positive and negatives instances presented, the kind of activity of identification, the amount of information given by the feedback, the nature of the prompts given, the availability of the attributes defining the concept, and the availability of the concept definition itself. Combinations of these parameters allowed us to define nine didactic strategies. The choice of these strategies remains partially arbitrary since various other strategies might be designed by combining these parameters in other ways and other parameters might be identified.

The strategies are represented in a procedural form. Most of these strategies consist of presenting positive and /or negative instances of the concept, and proposing some activities such as identifying a new instance, discriminating two quadrilaterals, selecting the relevant attributes of the concept, and so on. These strategies are described in appendix and an instance of a dialogue driven by strategy 3 is presented.

The choice of a strategy is managed by rules represented as frames. Each frame contains three slots :

- the conditions, which describe when this particular strategy is selected;
- the conclusions, which contain the list of procedures to call for applying this particular strategy ;

- the guiding-rate, which is a quantitative appraisal of how directly the strategy transmits the knowledge to the student rather than leaving him to acquire this knowledge by himself. [1]

At the outset, the **condition parts were empty**. The learning function of PROTO-TEG aims precisely to discover these conditions. However, the tutor is initially able to adapt its strategy to the student by using a mechanism based on the "guiding-rate". This information is compared to a component of the student model called the "guiding-level". The value of this parameter varies according to a student's performances : it is increased if the student fails on the test and decreased when the student succeeds.

The selection mechanism classifies the rules by comparing their guiding-rate to the guiding-level : the first rule will be the rule with the guiding-rate closest to the guiding-level. If two rules have the same guiding-rate, they are randomly permuted. The first rule of this ordered set is then selected : its "conclusion" tag is used to define a new procedure which is then executed.

This simple adaptation mechanism insures the system has an initial level of efficiency (which appeared later on as too high for the purpose of our experiment; see section 5). If the student failed to learn a concept after 6 trials (6 strategy applications), the concept was abandoned and the next one was presented.

After having run the self-improving function, the rules selection mechanism is performed in two stages. In the first stage, the rules are still ordered by the same guiding-based mechanism. The only difference is that, if two rules have the same guiding-rate, one with conditions and one without, the rule with conditions is placed before the one without conditions. In the second stage, the first rule of the ordered set whose conditions are satisfied by the student model state is selected. A rule without conditions is considered to be a rule with satisfied conditions.

3.1.3 The student model

At the outset, PROTO-TEG has at its disposal only a **primitive student model**. We call it "primitive" because it contains information which is not structured. It contains the history of students' work. In most of the ITSs, the authors organize this information into a structure which possesses a meaning. They call this structure misconception, bug, cognitive state .... The meaning of this structure is important for the author but not for the system. For the system what is important is the function of this structure : to decide which didactic strategy may be selected. PROTO-TEG will try to organize this information according to its usefulness without regard to its meaning.

The information enclosed in the primitive student model is represented by a list of vectors. A new vector is added each time a student takes a test. The composition of each vector is shown in Table II .This primitive student model also includes the guiding-level, whose role in strategy selection has been explained above.

---

1 This guiding-rate has been determined arbitrarily by the designer by assigning points to each parameter defining a strategy and summing these points for each strategy. The resulting rate has only a ordinal value. The guiding-rate assigned to each strategy is given in appendix .

Table II : Structure of each vector composing the primitive student model

Position	Element
1.	The number identifying the last strategy used ;
2 &3.	Number of strategies used since the beginning of the session (may be > 10);
4.	The number of strategies used to teach the current concept (≤6);
5.	The number identifying the current concept ;
6.	The guiding-rate of the last strategy used ;
7.	The number of under-generalization errors during the test ;
8.	The same number dichotomized (0 if no error,else 1) ;
9.	The number of over-generalization errors during the test
10.	The same number dichotomized (0 if no error, else 1) ;
11.	The score obtained by the student on the test ;
12.	The same number dichotomized (0 if the score is less than 9/10, else 1)

3.2 The Self-improving function

Our approach in designing the PROTO-TEG self-improving function has been largely inspired by the LEX program (Mitchell, Utgoff and Banerji, 1982). This program and several similar ones - reviewed by Langley (1983) - try to acquire or refine problem solving heuristics by analysing their solution path. They generally proceed in four steps :

1º) One or several problems are solved by applying rules. The right hand side of the rule describes some domain-specific operator. The left hand side describes when this operator can be applied (legal conditions) but does not insure that this operation will lead to a solution.

2º) When the solving process is completed, the system examines its solution process. "Problem state - Operator" pairs lying on the solution path are classified as positive instances of the heuristic to be discovered, the other pairs being classified as negative instances.

3º) Induction- and/or discrimination-based concept learning methods are applied to both instances sets in order to discover or refine heuristics. The learning methods used here differ considerably from one program to another.

4º) The left hand part of the rules are updated with the new heuristics and the entire process is repeated.

The same sequence characterizes the PROTO-TEG self-improving function (performed for each strategy):

1º) PROTO-TEG teaches a sample of students. During this experience, each time a strategy is applied, the state of the primitive student model is recorded. PROTO-TEG's didactic strategies correspond to LEX operators.

2º) This set of records is divided into two subsets : the records preceding a successful application of the strategy (i.e. if the student correctly identifies at least 9 out of the 10 quadrilaterals presented in the test) and those followed by a failure of this strategy. The former are considered as positive instances of the conditions to be found and the latter as negative instances.

3º) PROTO-TEG uses concept learning methods to learn from these two sets of instances. These methods are described below.

4º) The tutor rules are updated and the system is ready to be used again. In our experiment, it will actually be used by the same students so that we can perform some comparisons (see section 5).

The PROTO-TEG domain of learning differs significantly from the domains exemplified by these systems, which led us to develop specific learning algorithms. The two main differences are the presence of noise and the issue of choosing the right descriptors for expressing conditions.

Any learning mechanism confronted with real data meets the issue of different kinds of noise, especially when the data concern human behaviour. Our algorithms focused on one particular kind of noise, specific to this learning function : misclassifications. We know that experimental pedagogy never gives rules such as "This strategy is efficient for all (100%) the students who ...". There are always exceptions, many exceptions ! In other words, a particular strategy S may succeed with one student and fail with another student, both being described by exactly the same student model state. This means that this student model state will be present as a positive and negative instance of the same strategy S. Solving this problem requires tolerating exceptions among the positive and/or the negative instances.

Hence we designed learning methods that take exceptions into account : a characteristic will be considered as an attribute of the concept if it is common to a certain percentage of the positive instances (e.g. 80% of them) and absent from the same percentage of the negative instances. We will call **PE** the maximum **p**ercentage of **e**xceptions tolerated by the learning method. This parameter has been empirically determined by the experimenter.[2]

The second main difference between LEX and PROTO-TEG learning domains is that, in symbolic integration (LEX), the success of an operator is strictly related to the current problem state. But, the success of a PROTO-TEG strategy depends on the complete student history. This raised the problem of choosing attributes for describing this historical dimension. The first solution (first learning method) takes into account the chronology of events. For instance, it aims to discriminate events which happened during the previous strategy application from events which happened 5 strategies before. This approach is carried out by searching for concepts specific to one strategy application. Each strategy application being represented by one student model vector, the learning method will include a specific search for each vector.

The second learning method summarizes the student's history by counting the frequencies of events along this history, i.e among the complete list of vectors of each instance. Both methods are briefly described here. They include an induction and a discrimination phase. The application of these methods has been preceded by running procedures which identify the various instances of each strategy in the recorded files, classify them and adjust their length. Remember that an instance (a piece of the primitive student model) is a list of vectors composed of 12 elements.

---

2 This PE parameter might also be chosen by the self-improving function, starting with PE = 0 and increasing progressively PE until it finds concepts.

The **first method** determines if, for a place P, in a vector W, there is an element common to (100-PE)% of the number of positive instances. It may for instance discover that the first element of the last vector of 80 % of the instances is "3" (which means that, in 80% of the cases, when the considered strategy was successful, it has been applied immediately after the strategy 3). This operation is performed by a set of embedded recursive procedures which count the frequency of each value for each place of each word and retain the values whose frequency is higher than (100-PE)%.

Generalization is encompassed within the data since some elements are more general than others : the guiding-rate of a strategy is for instance more general than the number of the strategy, since several strategies have the same guiding-rate.

Specialization is obtained by searching for those combinations of *n* various induced elements which characterize (100-PE)% of the positive instances. It may for instance discover that the value "3" for the first element and "1" for the last element are simultaneously present among 80% of the positive instances (which means that, in 80% of the cases, when the considered strategy was successful, it has been applied immediately after a success of the strategy 3). This operation is performed by a depth-first search procedure corresponding to the following instantiated algorithm :

```
To Specialize { a /  b c d ...}
        IF { a b } characterizes (100 - PE)% of the positive instances
                        THEN record {a b} and specialize {a b /  c d ... }
        Specialize { a / c d .... }
        Specialize { b / c d .... }
End
```

Specialization makes the temporary elaborated concepts more resistant to the discrimination phase. This consists of rejecting the temporary concepts which characterize more than PE% of the negative instances. The resulting concepts form the disjunctive conditions for the considered rule : this rule will be selected if any of the disjuncts is satisfied.

The **second method** determines if, for a place P, there is a distribution of the values in the different vectors of an instance, which characterizes (100-PE)% of the positive instances and less than PE% of the negative instances. This method first processes each instance : the list of vectors is replaced by the list of the values on the place P in each vector. Then each list of values is replaced by a distribution indicating the frequency of each value in the list. For instance, the list [3 3 5 5] will be replaced by the distribution { [3 : 2] [5 : 2] [12 : 0] }. Finally, these distributions are compared and the program searches for frequencies intervals which includes the frequencies found in the positive instances and exclude those found in the negative instances. This method may for instance discover that, in 80% of the instances, the guiding-rate takes the value 3 between 2 and 6 times, the value 5 between 0 and 2 times and never the value 12 :

Position 6 (guiding-rate) : (3 : 2 -> 6) (5 : 0 -> 2) (12 : 0 -> 0).

In other words, this example means that the considered strategy was efficient when the student was used to a low level of guiding (often 3, sometimes 5 and never 12).

The search space for these concepts may be represented by a tree partially represented in figure 1. This three root is the initial temporary concept formed with the frequencies observed on the first instance. Each node represents the confrontation of this temporary concept with either a positive or a negative instance. The temporary concept is said to be consistent with a positive instance if the frequency of each value in the instance is inside the interval of frequencies for the same value in the concept. Consistency with a negative instance means that the frequency of each value in the instance is outside the frequencies interval of the same value in the concept. [3]

If the concept is consistent, then it is compared to the next instance (left branch). In the opposite case, the concept must be adapted to the instance (right branch) or this instance must be considered as an exception, i.e. the instance will not be taken into consideration by the concept (central branch). Adapting the concept to a positive instance is performed by enlarging the frequencies interval in order to include the frequency found in the instance. Adapting the concept to a negative instance is perform by removing the value-interval pairs which include the frequency found in the instance.

This search space is explored with a depth-first search method focusing on one temporary concept at a time until it takes into account all the positive and negative instances. If the concept is inconsistent with the instance, the reaction "adapt the concept" is systematically explored before the reaction "consider the instance as an exception". This heuristic allows us to minimize the number of instances being considered as exceptions.

Two situations start the backtracking process. The first one occurs when the number of examples considered as exceptions passes beyond the maximum percentage (PE%) allowed by the experimenter. The second situation happens when successive adaptations of the temporary concept lead it to be tautological : when all the value-interval pairs have been removed in order to adapt the concept to negative instances or because the interval covered all the possible frequencies (e.g. 0 to 5 when the instances maximally contain 5 vectors).

---

3 This condition is too strict - we might state consistency when at least one instance value is outside the concept interval - but is consistent with the disjunctive use of the conditions in the selection process.

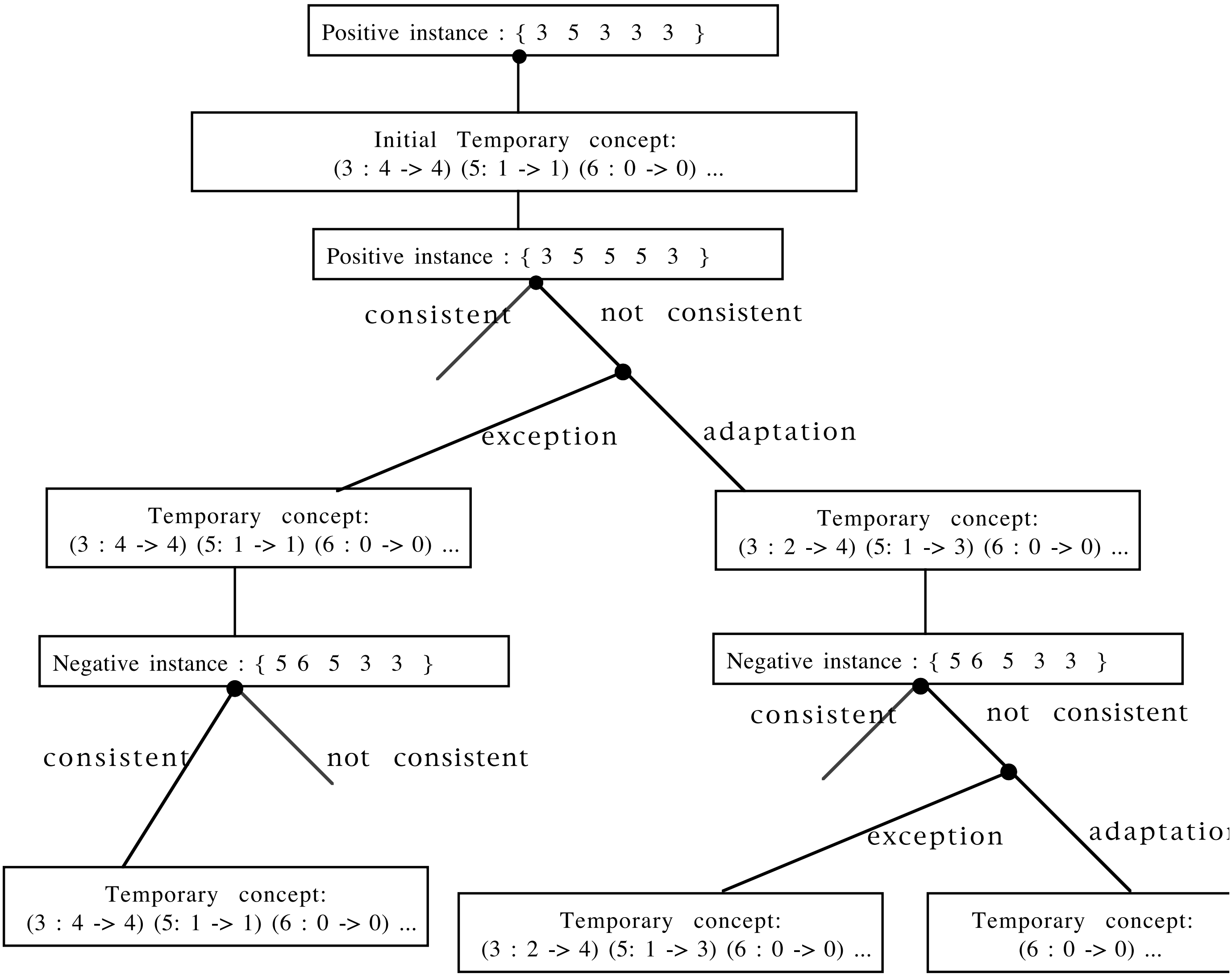

Figure 1 : Part of the tree representing the concepts search space by the second learning method.

4. Experimentation

4.1. Research questions

Two research questions arose from this work :

1. Would PROTO-TEG discover any conditions for its strategies ?

2. If so, would this knowledge allow the tutor to become more efficient, in terms of student's average scores on the tests ?

The first question aims to assess the power of the learning mechanisms, i.e. their ability to find some premises. The second question concerns the validity of the acquired knowledge. We use this measure of performance results for assessing the knowledge validity because the same measure (test scores) has been chosen for discriminating positive and negative instances.

4.2. Sample

Thirty undergraduate students in psychology were randomly divided into two groups of 15. The choice of undergraduate students rather than children enabled us to adopt minimal solutions during the interface design, but contributed to the appearance of a ceiling effect during the experimentation (see 5.7). Two students were withdrawn from the sample because the screen scale[4] had been accidently modified so that the presented figures did not correspond to the tutor knowledge.

4.3. Pilot experiment

A pilot experiment was carried out with three subjects in order to assess the understandibility of the task and evaluate the learning time. This led us to bring minor changes to the system, to reduce the number of concepts to teach and divide these concepts into two learning sessions (see section 4.1.1)

4.4. Experimental setting

For answering the first research question (Will PROTO-TEG discover any conditions ?), we had simply to use PROTO-TEG with our sample and then run next the self-improving function. Nevertheless, the answer to the second question required the elaboration of a experimental design presented in figure 2.

Two groups of students learned the first set of 4 concepts with the initial form of PROTO-TEG, i.e. with empty conditions in rules. After this learning experience, the learning function was applied and the discovered conditions were added to PROTO-TEG's tutoring rules. Then, both groups underwent a second learning session, including 4 different - but similar - concepts. The experimental group used the updated form of PROTO-TEG while the control group continued to use the initial form (without conditions).

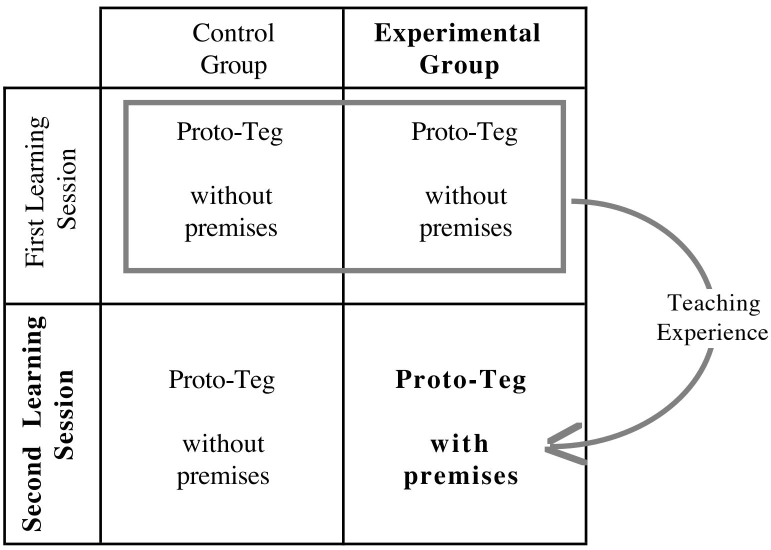

Figure 2 : Experimental design

4 This scale is the rate between the size of a vertical and an horizontal pixel. Its modification modifies lengths and angles.

We thus have two independent variables, "group" and "session" (plus the error term, the variable "subjects"). The dependent variables measure the increase in the tutor's efficiency in two ways :

- Score gains : improvement is expressed by the difference between the average score during the first and second learning sessions. This average score is the sum of the different scores during one session, divided by the number of tests. This gain (or loss) may vary between + 10 and - 10.

- Number of strategies : another way to compare efficiency is to see if the average number of rules for learning a concept has decreased. The average number of rules is obtained by dividing by 4 (the number of concepts per session) the number of strategies applied during one session.

4.5 Global evaluation of the tutoring part.

Although assessing the tutoring system was not the main object of this research, this experiment taught us that PROTO-TEG was very efficient with this sample (89.6% as mean score for the control group during the first session). Nevertheless several criticisms were expressed by the subjects.

Several students emphasized the difficulty of making measurements on the screen, especially length measurements. Other students complained about the slowness of the quadrilaterals producer, especially for negative instances. Finally, one student found a logical gap we had not anticipated : all the BULI instances he received had no right angle, hence, he correctly induced this negative attribute as a characteristic of the concept ! These criticisms all concern the quadrilaterals producer. They might be avoided by replacing this generative process by a data-base of quadrilaterals accessible through their characteristics.

We also found from the experiment that the second learning session, although composed of very similar concepts, was significantly more difficult than the first learning session, as shown in table III. This difference was probably related to the fact that the second session concept required more length measurements (quoted as difficult by students).

TABLE III : Comparative difficulty of learning sessions 1 and 2.

Data from the control group	First learning session	Second learning session	Test T of Wilcoxon (n=14)
Average score on the test	8.96	8.44	T = 21 ; p = .05
Average number of strategies	1.26	1.93	T = 3 ; p = .01

4.6 Results

The answer to the first research question is partially positive : by applying its self-improving function to its tutoring experience, our system discovered premisses for five of the nine didactic strategies, with the first learning method. The percentage of instances considered as exceptions varied between 15% and 25%. Table IV summarizes the discovered premises. The complete results are presented in appendix.

Table IV shows us that the second method failed to learn any concept up to 25% of exceptions. We have not been able to try this method with a percentage of exceptions greater than 25% because of overflow problems. Nevertheless, the first learning method succeeded to learn at the same threshold. An explanation of this failure seems to be that, during the discrimination phase, we have considered that all (rather than one) concept characteristics must be absent from negative instances.

Table IV: Conditions discovered by PROTO-TEG learning methods

	Number of discovered conditions			Examples of discovered conditions
	First Learning Method		Second Learning Method	
	Uni-dimensional concepts	Multi-dimensional concepts		This strategy is efficient ...
1	Insufficient number of instances			—
2	3	8	0	If it is the 4th strategy after a test success
3	2	10	0	If it follows strategy 5
4	0	14	0	If the student learned the previous concept on her first trial
5	0	0	0	—
6	Insufficient number of instances			—
7	0	0	0	—
8	1	1	0	If the student succeeded the previous test
9	2	4	0	If it is the 7th strategy after a test where the student did not make any over-generalisation mistake

The first learning method has been more successful. We may note that when it found some results, it generally found many of them. Nevertheless, most of these results where partially redundant[5]. Some of the concepts obtained were meaningful. For instance, PROTO-TEG discovered that strategy n°3 is more efficient if it follows strategy n°5. This result is explicable since strategy 5 presents the students with the various attributes available to build the concept, which facilitates the subsequent learning. Another understandable premiss discovered stated that learning with strategy n°4 (the most difficult) was efficient if the student succeeded in learning the previous concept on the first trial, i.e. if the learner has already attained a high level of competence.

Other PROTO-TEG discoveries are more difficult to explain : for instance, PROTO-TEG has established that strategy n°9 is efficient if it is the seventh

5 If the result is a set of conditions {a,b,c,d} and if a=>b and c=>d, then a and c are redundant, the set may be resumes to {b,d}

strategy to be used after a test in which the student did not make an over-generalization mistake ! This kind of result is perhaps a consequence of the smallness of the sample used in the experiment.

The answer to the **second question** is negative. Table V shows a slight improvement by the fact that the score decrease between the two sessions was lower in the experimental group. Nevertheless, given our reduced sample, these differences may not be considered as statistically significant.

4.7 Discussion

The first explanation for the absence of improvement is that the acquired knowledge was not valid. Several arguments support this explanation.

First, the input of the learning mechanisms was quantitatively and qualitatively poor. The quantitative issue refers to the limited size of the sample, i.e. to the limited teaching experience, especially if we take into account the percentage of exceptions we have been constrained to tolerate. The qualitative problem is that the primitive student model used as learning input was too simplistic, mainly composed of qualitative parameters. This raises one fundamental paradox in the research on ITSs. In order to get some results in the short term, we must focus on one particular component of an ITS and adopt less-than-optimal solutions for the other components. But, because all components are interdependent, this simplification later indirectly limits the performance of the component in focus.

Table V: Measures of self-improvement

Mean of gains between the two learning sessions	Control Group			Experimental Group			Test t of Student (n=28)
	Pretest	Post-test	Diff.	Pretest	Post-test	Diff.	
Average score on the test	8.96	8.43	-0.53	9.03	8.56	-0.47	t= 0.23 not significant
Average number of strategies	1.27	1.93	0.67	1.25	1.65	0.40	t = 1.25 not significant

Another source of lack of validity for the knowledge acquired is faults in the learning mechanisms themselves. Both learning methods used very simple forms of generalization during the induction process, and, in the second method, the discrimination step eliminated too many concepts.

On the other hand, the absence of improvement may also be attributed to some factors external to the learning process itself. First, PROTO-TEG has an initial efficiency of 84% which limited the improvement (ceiling effect). This efficiency was due to the easiness of the learning task for the chosen sample and to the effectiveness of the initial guiding-based adaptive mechanism. Secondly, the condition-based adaptive mechanism was subordinated to the guiding-based mechanism since the later classified the rules before the former checked their conditions. Hence, the improvement effect expected from adding a (partial) condition-based mechanism was in some way hidden by the prevalent and efficient guiding-based mechanism. Thirdly, the differences between the didactic strategies was too small ; the choice of one strategy rather than another did not constitute a change important in the student activities. Finally, the ability to measure attributes on the screen was a bias in assessing student's mastery of the concepts : student might have acquired the correct concept but fails to measure correctly the instances presented in the test.

5. Concluding remarks

The PROTO-TEG implementation corresponds to an attempt to explore the self-improving abilities of a tutoring system. This system may not be considered as a complete intelligent tutoring system. Its role was to show that it is possible to design a tutor able to learn when to select a didactic strategy. More precisely, PROTO-TEG aimed to build categories of student model which predict strategy efficiency. These categories are defined as data structures which differentiate the primitive student model states recorded before successful applications of a strategy and those recorded after a failure of this strategy.

As with every learning system, the knowledge that PROTO-TEG is able to acquire is strongly determined by the knowledge it has initially. The attributes and values used for building the concept are encompassed in the primitive student model and in the learning mechanisms themselves. Further investigation of the heuristic power of this prior knowledge should be required for progressing in this field.

We have approached the problem of dealing with noisy data (exceptions) by using two different learning methods. The first one produced some interesting results, but the second was not successful. This issue and the previous one (background knowledge) remain major challenges for researchers in machine learning.

We have been confronted with a contradiction in this work : a tutor is constrained to learn quickly, for obvious ethical reasons, but, on the other hand, it needs a large teaching experience to guarantee the validity of its learning products. One clear limitation of PROTO-TEG was its inability to discriminate valid acquired knowledge from results due to chance.This problem might be solved by comparing different sources of confidence in the acquired knowledge : when the statistical evidence, resulting from some similarity-based learning method (as in PROTO-TEG) is lacking, the tutor should attempt to find theoretical evidence by using explanation-based learning methods (Mitchell et al., 1986)). This method requires that the system can map the acquired conditions to an explicit description of the didactic strategy in order to "explain" their relationship. This requirement led us to propose the concept of reflective tutoring system as a generalization of the concept of self-improving system, the improvement being considered as a particular result of reflection (Dillenbourg and Goodyear, 1989).

Acknowledgments.

We want to thank Prof. D'Hainaut and Dr. C. Depover who supervised this work, Dr. P. Goodyear for several discussions on self-improvement and Dr. J.A. Self and Dr. S. Ohlsson for their comments on successive drafts of this paper.

References.

DILLENBOURG P. (1988) Un modele d'acquisition des connaissances par les tutoriels intelligents. Proceedings of Intelligent Tutoring Systems. June 1-3. Montreal. pp. 145-153.

DILLENBOURG P. and GOODYEAR P. (1989) Towards reflective tutoring systems : self-improvement and self-representation. In Bierman D., Breuker J. and Sandberg J. Artificial Intelligence and Education. Proceedings of the 4th AI&Education conference. IOS, Amsterdam.

DUCHASTEL, P. and IMBEAU J. (1986) Instructible ICAI. Doc. 86-07 du Laboratoire d'Intelligence Artificielle en Education. Universite Laval. Quebec.

KIMBALL R (1982) A self-improving tutor for symbolic integration. *In* : SLEEMAN D. & BROWN J.S (Eds), Intelligent Tutoring Systems. Academic Press.London

HARTLEY J.R. and SLEEMAN D.H. (1973) Towards more intelligent teaching systems. International Journal of Man-Machine Studies, vol.5, pp.215-236.

HOWE J.A.M. (1978) Artificial intelligence and computer-assisted learning : ten years. Programmed learning and Educational Technology.Vol.15, nº2, pp.114-125.

LANGLEY P (1983) Learning search strategies through discrimination. International Journal of Man-Machine Studies, 18, 513-541.

LANGLEY and OHLSSON (1984) Automated cognitive modelling, Proceedings of National Conf. on Artificial Intelligence. Austin, Texas, pp 193-197.

MITCHELL T.M., UTGOFF P.E and BANERJI R. (1984) Learning by Experimentation : Acquiring and refining problem solving heuristics. In : Michalski R.S., Carbonell J.G and Mitchell T.M (Eds) Machine Learning. An artificial intelligence approach. Springer-Verlag, Nwe York, pp. 163-190.

MITCHELL, T.L., KELLER, R.M., and KEDAR-CANELLI S.T. (1986) Explanation-based learning : A unifying view. Machine learning, (1), 47-80.

OHISSON S. (1987) Some principles of intelligent tutoring. *in* Lawler & Yazdani, Artificial Intelligence and Education. Volume one. pp 203-237.

O'SHEA T. (1979) Self-Improving tutoring systems. Birkhauser Verlag, Basel.

SELF J.A . (1977) Student models and artificial intelligence. Computers and Education, Vol.3, pp.309-312.

SELF J.A. (1986) The application of machine learning to Intelligent Tutoring Systems *in* Self J.A. (Ed) Intelligent Computer Aided Instruction, London, Chapman and Hall.

SELF J.A. (1988) Bypassing the intractable problem of student modelling. Proceedings of Intelligent Tutoring Systems. June 1-3. Montreal. pp.18-24.

STUBS M. and PIDDOCK P. (1985) Artificial intelligence in teaching and learning : An introduction. Programmed learning and Educational technology., vol.22,nº 2, pp.150-157.

TENNYSON R.D. and PARK O. (1980) The teaching of concepts : a review of instructional design research literature. Review of Educational Research, 50, pp. 50-70.

APPENDIX

Description of the didactic strategies used by PROTO-TEG :

Strategy 1 : - presents 3 positive and 2 near-miss negative instances ;
- presents a quadrilateral and asks whether it is an instance or not ;

- gives a feed-back ;
- uses the near-miss instances in the negative feed-back ;

Guiding-rate : 6

Strategy 2 :
- presents 3 positive and 3 negative instances (including two near-miss ones);
- presents a list of attributes and asks to select the relevant ones ;
- gives a feed-back ;
- uses the near-miss instances in the negative feed-back ;

Guiding-rate : 12

Strategy 3 :
- presents 2 positive and 2 near-miss negative instances ;
- present 2 quadrilaterals and asks which one is an instance ;
- gives a feed-back ;
- uses the near-miss instances in the negative feed-back ;

Guiding-rate : 6

Strategy 4 :
- presents 5 positive instances ;
- presents a quadrilateral and asks whether it is an instance or not ;
- gives a feed-back ;

Guiding-rate : 3

Strategy 5 :
- presents 5 positive instances ;
- presents a list of attributes and asks to select the relevant ones ;
- gives a feed-back ;

Guiding-rate : 9

Strategy 6 :
- presents 4 positive instances ;
- presents 2 quadrilaterals and asks which one is an instance ;
- gives a feed-back ;

Guiding-rate : 3

Strategy 7 :
- presents 5 positive instances ;
- presents a quadrilateral and asks whether it is an instance or not ;
- gives a feed-back ;
- in the negative feed-back, gives a prompt on the nature of one of the attributes ;

Guiding-rate : 5

Strategy 8 :
- presents 5 positive instances ;
- presents a quadrilateral and asks whether it is an instance or not ;
- gives a feed-back ;
- in the negative feed-back, tells the number of attributes in the concept's definition ;

Guiding-rate : 6

Strategy 9 :
- presents the definition of the concept ;
- presents an instance ;
- shows the presence of the attributes on the instance ;

Guiding-rate : 15

Complete Results of the first learning method.

Strategy	Places	Concepts
2	7	A X X A X A
	10	B X X A X X

	12	0 X X X X X X
	2 & 7	AA X X AA X X
	2, 7 & 8	AAA X X AAA X X
	2 & 10	AB X X AA X X
	2, 11 & 12	AI0 X X X
	2, 12	AO X X X X X X
	7 & 8	AA X X AA X AA
	8 & 12	BO X
	11 & 12	IO X X X
3	1	5
	6	G J
	1 & 2	5A
	1,2 & 6	5AJ
	1,2,6 & 12	5AJ1
	1,2 & 12	5A1
	1 & 6	5J
	1,6 & 12	5J1
	1 & 12	51
	2 & 6	AG AJ
	2, 6 & 12	AJ1
	6 & 12	J1
4	2, 4 & 7	AAA
	2,4,7 & 8	AAAA
	2,4,7,8 & 12	AAAA1
	2,4,7, & 12	AAA1
	2,4 & 8	AAA
	2,4,8, & 12	AAA1
	2,4 & 12	AA1
	4 & 7	AA
	4,7 & 8	AAA
	4,7,8 & 12	AAA1
	4,7, & 12	AA1
	4 & 8	AA
	4,8, & 12	AA1
	4 & 12	A1
8	12	1
	2 & 12	A1
9	9	A X X X X X X
	10	A X X X X X X
	2 & 9	AA X X X X X X
	2,9 & 10	AAA X X X X X X
	2 &10	AA X X X X X X
	9 & 10	AA X X X X X X

A = 0 , B = 1, I = 8 , the X are used to indicate the chronological position, counting backwards from the strategy considered. The signification of positions are given in section 3.1.3

We present here an instance of a dialogue driven by **strategy 3.** This dialogue is presented by screen snapshots. We must emphasize that ratio between the breadth and the height of a dot was not the same on the screen and on the printer and that some angles consequently have been modified by printing. We nevertheless present these screens because they give an idea of interactions between the learner and the system. The arrow shows the place where the student introduced his / her response.