KNOWLEDGE ACQUISITION BY OBSERVATION: APPLICATION TO INTELLIGENT TUTORING SYSTEMS

Guy A. BOY and Nicole NUSS
Groupe d'Intelligence Artificielle
Centre d'Etudes et de Recherches de Toulouse
Office National d'Etudes et de Recherches Aérospatiales
2, avenue Edouard Belin
31055 Toulouse Cedex
France

Abstract. This paper presents an architecture for an Intelligent Tutoring System (ITS) based on a pedagogic model. This model takes into account the student's behavior in a "short term supervisory control loop" and improves the initial pedagogic knowledge base in a "long term evaluation loop". Observation is the key factor in this approach. It allows incremental knowledge acquisition from an initial pedagogic knowledge base. A learning technique by Specialisation-Structuring has been developed and is presented here. This approach is being applied in instruction of aircraft pilots.

Keywords: Intelligent Tutoring Systems, Knowledge Acquisition, Observer, Pedagogic Model, Complex Environment, Global Approach.

1. Introduction

Intelligent Tutoring Systems (ITSs) are programmed teaching-aids grounded in artificial intelligence (AI) (Sleeman & Brown, 1982). AI techniques can be used for improving computer-assisted instruction in three areas: (1) improving student-computer interaction, (2) mastering domain knowledge through problem solving, and (3) taking into account the student's model and understanding of his/her behavior. Our work is more focused on area 3. Conventional tutoring systems use very little pedagogic knowledge and need to improve the modularity of their knowledge bases (KBs) (Forcheri & Molfino, 1986). The major critical problem in KB systems development is the knowledge acquisition process (Freidland, 1981; Feigenbaum, 1982; Freiling et al., 1985; Gevarter, 1984; Quinlan, 1984). This is particularly so in dynamic and moving environments. In the present knowledge acquisition technique, the initial KB includes technical and pedagogic knowledge. Technical knowledge is dependent on the domain to be learnt. Pedagogic knowledge has to be student-dependant according to on-line observation. It can be modelled as a meta level of the technical knowledge. As pedagogic knowledge acquisition is a major limiting step, it will be the focus of attention for the following technique.
With this more global approach, it is possible to tackle complex domains. For instance, our domain is flight management instruction. Our students are transport pilots. In this complex situation, the main goal is design a pedagogic model. This new approach in knowledge acquisition comes from a general concept of operator assistant (OA) systems (Boy, 1988a). OA systems are both computer aids and observers of the environment, including the system being controlled and the human operator himself. In the present work, we will talk about "student assistant system". This paper presents a pedagogic model which takes into account two types of information acquisition from the environment. On the one hand, the short term supervisory control loop allows the system to react to observed student behavior, i.e., to correct immediate errors. On the other hand, the long term evaluation loop allows modification of the knowledge base from observation and interpretation of the obtained results. As the topic of this paper is concerned with intelligent tutoring systems and "expert system" knowledge acquisition, we will differentiate between human learning and machine learning to avoid any confusion.

2. A Pedagogic Model

Good professors are also good observers of the behavior of their students. They are able to analyze and propose adapted pedagogic situations to offer their students an easier problem to solve. This constitutes the main issue in the present approach. Our pedagogic model (Figure 1) includes two loops: (1) a short term supervisory control loop, and (2) a long term evaluation loop.

The Short Term Supervisory Control Loop

The first loop of the pedagogic model concerns the supervisory control of the human learning process. The student is assumed to be faced with a learning situation. He has to try actions or to answer questions from the ITS. Thus, a situation can be generated by (1) the environment, including questions, menus and actions to be taken during training simulations, and (2) the pedagogic knowledge involved to create such a situation. A real situation is defined as a set of facts of the world related by logical operators. For instance, a real situation is to learn how to use the flight management system on an aircraft. Such a situation is very difficult to perceive by a human being when he is not well trained (which is the case in a training context). Thus, the student has to build a set of situation patterns (Boy, 1987) to make him able to perceive the real situation as it is for better understanding. The human information processor has been shown to behave using two kinds of processing: (a) the situation recognition which is based on a pattern matching between the real situation and situation patterns already learned, and (b) the analytical reasoning which involves inference. The goal is then to facilitate the student's acquisition of "good" situation patterns.

The same applies to the ITS perceiving what the student is doing in a given situation. This task is allotted to the supervisory control observer module. The observed behavior is a projection of the real situation, including what the student is doing, on a grid called observation grid (Figure 2). This observation grid will be described later. The observed
situation is an input of the analyser. The analyser is a processor using the actual pedagigic and technical KB. Output of the analyzer module are sequentially processed by a proposer module which provides a proposition, i.e., either advice or a new pedagogic situation to act on. A pedagogic situation is an exercise designed to help correct previous misunderstanding and lack of knowledge. Generally, a pedagogic situation must be simple enough to be solvable by the student.

![Pedagogic Model](image)

**Figure 1 : Pedagogic Model.**

**Long Term Evaluation Loop**

The second loop of the pedagogic model is more specifically devoted to machine learning. The principle is almost the same as for the previous loop. The role of the learning-oriented observer module is to detect new situations for later analysis and possible machine learning. This observer includes the same set of patterns as the supervisory control observer. It is working on those patterns as a partial pattern-matcher, i.e., using analogy, and detecting new patterns by complementing the actual pattern base. Outputs of the observer module are processed by the evaluator module. The evaluator is a processor using an external knowledge base and the "intelligence" of
the system builder. This knowledge is methodological. Outputs of the evaluator are modifications of the existing pedagogic knowledge bases.

This loop is not tractable before implementation and testing of the first one. At present, this loop is open and directly regulated by the system builder, so it will not be described here.

![Figure 2: Observation Grid](image)

3. The Observer Model

In the following, a general observer model will be described for the short term supervisory control loop. However, this model can be used for the long term evaluation loop. The observer procedure works using objects to be filled in by events coming from the environment. An object is associated with each action of the student. A student action could be an answer or a selection. The behavior inducing each student's action is determined by procedural attachment. Thus, each action either generates a new object or modifies the content of existing objects.

3.1. The Observer Knowledge Base

The goal is to assess the performance of the student. The corresponding knowledge is a set of students' actions. The observer KB or observation grid is modelled using an object-oriented representation. In particular, each student's action is perceived as an object.
The generic object template is the following:

Object associated to Action_A:
Classes: Exercice_N
Problem_Type_M
Properties:
  Action Description: Question_Expression
  Input_Device [mouse, keyboard, ...]
  Effective_Action
  Previous_Attempts: Number
  {Effective_Action(s)}
  Truth_Value: [True, False, Not_Known]
  Action_Type: [Suggested, Voluntary]
  Elapsed_Time: From_Beginning_of_Session
                 From_Beginning_of_Exercice
                 From_Last_Input
  Student_Belief: [I_know, I_try_to_see, I_dont_know]
  Behavior: Procedural_Attachment
            -> Try_and_Error
            -> Slip
            -> Processed_by_Logic
            -> Processed_by_Analogy
  Weight: [0..100]
  Student_Level: [Beginner, Medium, Advanced]

An example of an objet is the following:

Object associated to action_4:
Classes: Exercice_Flight_Plan_Pages
Problem_Type_Page_Changing
Properties:
  action Description: Question_Expression:
    Go_from_Flight_Plan_Page_A_to_B
    Input_Device: Keyboard
    Effective_action: Slew_key_down
  Previous_Attempts: Number: First
  Truth_Value: False
  Action_Type: Voluntary
  Elapsed_Time: From_Beginning_of_Session: 9'34
                 From_Beginning_of_Exercice: 3'02
                 From_Last_Input: 4'
  Student_Belief: I_know,
  Behavior: Procedural_Attachment
            -> Processed_by_Analogy with
            Go_from_Init_Page_A_to_B
  Weight: 12
  Student_Level: Beginner
Interesting behavioral entities are sequences of actions rather than single actions. Automatic detection of such sequences is a really tricky problem. One way to approach this is to elicit sequence patterns by experimentation. An action sequence pattern is an object defined as follows:

Action sequence pattern:
- Valid_Contexts: {list of valid contexts}
- Action_Sequence: {list of actions}

In this paper, we will take the following notation:

\[ \{(C_1, C_2, ..., C_n), (A_1, A_2, ..., A_m)\}, \]

where \( C_i \) is a context and \( A_j \) is an action.

Contexts represent the field of application of the corresponding actions. Contexts and actions, i.e., patterns, are built incrementally after compiling several experiments with students.

### 3.2. The Observer Processor

The observer processor works as presented in figure 3.

**Pattern Matching.**

After each student action, the processor filters this action through a "relevant" sub-set of action sequence patterns. Such a relevant sub-set is preselected from the whole knowledge base according to the present context. Retrieval criteria will not be described here. The context filter works taking into account the actual situation. It selects a sub-set of patterns called *active* action sequence patterns. Contexts are predefined by the system builder taking into account a priori technical and pedagogic knowledge. In the next research phase, contexts will be updated by the long term evaluation loop.

The pattern matching can be complete or partial. It is complete when every action in the predefined sequence have been performed by the student. Given the action sequence \((a_1, a_2, ..., a_m)\) valid in the context \(c\),

\[
\text{IF } \{a_1=A_1; a_2=A_2; ...; a_m=A_m\} \text{ and } c \text{ is member of } \{C_1, C_2, ..., C_n\}, \\
\text{THEN the pattern matching is complete.}
\]

However, a pattern can be "partially" matched if a reasonably number (taking into account action weights) of "important" actions have been completed. In all cases, several patterns could be matched. Given the action sequence \((a_1, a_2, ..., a_p)\) valid in the context \(c'\),
IF \{a_j=\text{A}_s(j) \} / where j is member of \{1, 2, ..., p\},
where p < m
and s is an application from \{1, 2, ..., p\} into \{1, 2, ..., m\},
AND c' is member of \{C_1, C_2, ..., C_n\},
AND \{ \prod (w(a_j)) \geq T \} / where j is such as a_j=\text{A}_s(j),
where \prod is a conjunctive formula to be defined,
and w is a weight function from (a_1, a_2, ..., a_p) into [0, 1]
and T is a threshold,
THEN partial pattern matching between (a_1, a_2, ..., a_p) and the action sequence pattern (A_1, A_2, ..., A_m).

Conflict Resolution.

Conflict resolution is the process which will select one or several patterns out of the matched ones. Rules for such process is very domain dependant. Rules can select either one hypothesis taking into account a given criterion, or several hypotheses for later parallel processing. Present implementation is based on a single hypothesis selection. For each completely or partially matched sequence, weight factors are combined through a conjunctive formula such as defined above for partial pattern matching. Results of such conjunctive formulae applied to each matched pattern are compared together and the best is then selected.

Inference.

When an action sequence pattern is selected, a relevant hypothesis is suggested to the diagnoser. As this system is connected with a dynamic environment, i.e., erratic actions of the student, patterns can be fired even when the diagnoser is working. Such characteristics impose on the diagnoser the necessity to be able to revise its beliefs on facts it has already deduced. In other terms, if the level of the student has been deduced as "advanced" and an action sequence pattern suggests that his/her level should be "beginner", then a belief revision must occur. Thus, the diagnoser includes two parts: a problem solver and a belief maintenance system. The role of the problem solver is to deduce facts from initial facts and rules in order to prove a selected relevant hypothesis. The belief maintenance system can revise truth and values of facts according to a new situation. This process is performed by resetting rules already used. Such a feature allows incremental reasoning in changing environments.
4. The Knowledge Acquisition Process

In our approach, the main difficulty in building the system is to design situation patterns, i.e., action sequence patterns, and chunks of analytical knowledge, i.e., diagnoser rules. Such knowledge will be acquired using an adapted acquisition procedure. At present, this procedure is performed manually. However, it will be described formally in the following.

4.1. Classical methods

There is no universal pedagogic knowledge. The difficulty in such knowledge acquisition (KA) is the variability of sources. Moreover, a given pedagogy can be very efficient with one student and very bad with another. Thus, student reactions to pedagogic inputs must be observed and processed in order to correct initial pedagogic strategies.

The first approach in the KA process is to enter into the pedagogic knowledge base a presumably "good" sub-optimal knowledge. This first approach includes the following classical methods of manual KA (Kidd & Welbank, 1984): interviews of tutors (tape-
recorded, structured (Lafrance, 1986), or using interview tools (Boose & Bradshaw, 1986), brainwriting (Warfield, 1971), brainstorming, direct observation (Boy, 1986), and verbal protocol analysis (Ericsson & Simon, 1980).

The second approach can be characterized by building knowledge design interfaces (Rappaport, 1986). Such interfaces are accessible and user-friendly to the tutor him/herself. They are designed for the tutor to enter his/her own pedagogic knowledge according to each significant observed situation. This approach reduces bias introduced by a knowledge engineer. For example, good tutoring systems have been shown to be built by computer scientists themselves for computer science teaching. The only two systems which are commercially available and in every day use are in the domain of computer programming: PROUST (Johnson & Soloway, 1985) and LISP tutor (Anderson & Reiser, 1985).

The last approach is machine learning (Michalski et al., 1983, 1986). It transforms an initial sub-optimal knowledge base into one which is both more general and better adapted to a particular student. Machine learning starts to be an interesting approach when initial knowledge bases are reasonably big, i.e., we learn very little when we know very little, and the converse. Machine learning is presently divided into two approaches: similarity-based learning (SBL) and explanation-based learning (EBL) (Kodratoff, 1986).

4.2. Building patterns

At the beginning of ITS construction, the KB is sub-optimal. Each situation pattern, for instance, is simple and relies on a complex analytical KB. For example, a delay on the reaction time of the student is an a priori situation pattern which implies a complex analysis to infer. Either he/she does not understand a system question or the overall situation itself. As the ITS is used, the system builder (who can be a tutor) becomes more knowledgeable about accurate and relevant observation patterns. Pattern construction is a top-down approach, i.e., patterns are designed from system-to-be-learnt specifications, and a bottom-up approach by experimentation, i.e., by observing student behaviors and reactions when faced with the ITS.

4.2.1. Aim of the Method

Learning can be seen as a two-directional transformation: specialization and generalization. Most of efforts in machine learning research concentrate on generalization, however as far as expertise is concerned specialization plays a primary role. Indeed, an expert is not necessarily "intelligent" in the sense of highly general sophisticated knowledge, but he has very domain-specific specialized patterns built from experience. Generalization consists of building disjunctive chunks, as opposed to specialization which consists of building conjunctive chunks.

The knowledge acquisition method presented here belongs to the class of EBL techniques. In the present method, explanation comes from experts as well as from
observers. Input data to the learning process is procedural knowledge. The knowledge acquisition process consists of three phases: (1) the gathering of procedural knowledge for construction of an initial sub-optimal KB, (2) the transfer (specialization) of elements of this analytical knowledge into situational patterns by experimentation, and (3) the Structuring of particular chunks of the analytical KB. To a certain extent, this method is similar to the chunking method developed in SOAR (Laird, Newell, & Rosenbloom, 1984). The difference is that here chunking effectively transfers chunks of knowledge from the AK base to the situational knowledge base. Situation pattern-oriented chunks permit parallel processing not only of the situational and analytical knowledge bases, but also several analytical reasonings simultaneously. As in SOAR, the search in the AK base is reduced because of the specialization process.

4.2.2. Knowledge Acquisition by Specialization/Structuring

Initial Sub-Optimal KB.

The first phase of the process consists of building a sub-optimal KB which will be refined by experimentation. The sub-optimal KB includes "small" rough situation patterns which are parameter-sensitive "demons" suggesting further analytical reasoning. The initial analytical KB describes how the system to be controlled works. It is split into a few sub-sets of rules, each of which includes large numbers of rules (Figure 4).

In our ITS project, an example of the original sub-optimal KB is:

SITUATION PATTERN:

IF { Time_spent > 45 sec. }  THEN suggest Hypothesis(A)
AND start-analytical-reasoning.

ANALYTICAL REASONING:
IF Pedagigic_KB_Validation THEN Hypothesis(A) 
   AND reset Pedagigic_KB_Validation.

This preliminary "analytical" rule works in backward chaining. The subsequent sub-set of rules is used generally in forward chaining and can be summerized in the following graph (Figure 5).

```
HYPOTHESIS-A
   LAST-STUDENT-ACTION
      RIGHT
      WRONG
         PREVIOUS-MODE
            DATA-DRIVEN
               CONTINUE
            GOAL-DRIVEN
               VERIFY-STUDENT-UNDERSTANDING
         HELP-CORRECT-MISTAKE
```

Figure 5 : Analytical Search Tree

Specialization.

It has been shown that as professors gain experience in teaching, their situation patterns become more complex (Figure 6), dynamic and numerous (Boy, 1987). In the example described above, it happened that all experts considered the node PREVIOUS_MODE in the early stages of the detection process. In this example, PREVIOUS_MODE could be either DATA_DRIVEN or GOAL_DRIVEN. Thus, this element of knowledge had to be traferred to the situational part of the KB. The specialisation of the KB has been carried out as follows.
Figure 6: Acquisition of Specialized Patterns

(1) Detection of the attributes or instances of the node to be transferred. Here, PREVIOUS_MODE can be DATA_DRIVEN or GOAL_DRIVEN.

(2) Identification of the initial situation pattern leading to this node and building a number "n" of new patterns corresponding to the number of values or instances of the transferred node. Each new pattern is a conjunction of the content of the initial pattern and each value or instance of the transferred node. In the above example, the new situation pattern is:

IF { Time_spent > 45 sec. 
    AND PREVIOUS_MODE (DATA_DRIVEN) } THEN suggest Hypothesis(A1) 
    AND start-analytical-reasoning.

IF { Time_spent > 45 sec. 
    AND PREVIOUS_MOVE (GOAL_DRIVEN) } THEN suggest Hypothesis(A2) 
    AND start-analytical-reasoning.
Figure 7: Specialization Process.

(3) Restructuring of the analytical KB. As n patterns have been created in step (2), the analytical KB must be split into n corresponding sub-sets. This transformation leads to duplication of the part of the analytical KB up to the transferred node and keeping the n branches corresponding to the instances of the transferred node. In a more schematic form, let T be the node to be transferred, I1 and I2 are particular values or instances of T, S is the initial situation pattern, and A,B,C,D,E are other nodes. The specialization process is given in Figure 7.

Structuring.

The Structuring process includes both analytic (explanation-based) learning on the analytical KB side and heuristic (similarity-based) learning on the situation pattern side.

Step (3) in the specialization process stresses the need to structure the part of the graph up to the transfer node. The sub-set corresponding to Hypothesis(A) will be called A. Structuring in the analytic KB introduces the concept of hierarchy among the sub-sets.
of rules. In the above example, sub-sets A1 and A2 have been built from the initial sub-set A. A1 and A2 both include the same sub-set of nodes (B and C). Thus, these nodes are general to the disjunct (A1 v A2). It follows that Structuring leads to the building of disjunctions of sub-sets of rules. The resulting structure of the analytical reasoning will be as follows. When a situation pattern is fired, an hypothesis is suggested. To prove this hypothesis, a general sub-sets of rules has to be activated. If these rules are satisfied, the reasoning continues in forward chaining.

![Diagram of Structuring Process](image)

**Figure 8 : Structuring Process**

The implementation of this technique in the form of rules leads to the first backward chaining rule (for A1 for instance):

\[
\text{IF Pedagogic\_KB\_Validation} \quad \text{THEN Hypothesis(A1)} \\
\quad \text{AND reset Pedagogic\_KB\_Validation} \\
\quad \text{AND suggest Hypothesis(A1vA2).}
\]

and the sub-set (A1 v A2) starts with a backward chaining rule of the same format. If a first backward chaining rule does not include a suggested action in its right hand-side, the corresponding sub-set of rules will be called a terminal sub-set. Otherwise, it will be a common sub-set. Each sub-set of rules has a given priority according to its degree of generality. This feature permits recursivity on several general sub-sets of rules. Note that a general sub-set of rules can be initiated directly by its own situation pattern, if there is one.
5. Discussion

Our first intention in this work was to develop a general theory for knowledge acquisition. The present attempt is a part of a wider research effort on operator assistant systems. Thus, we tried to tackle intelligent tutoring in this framework. In particular, the Situation Recognition Analytical Reasoning (SRAR) model has been developed (Boy, 1988b) and is being used as a representation reference for implementing student behavior observables. Indeed, observation is the key factor in this approach. As shown above, action sequence patterns are the limiting entities to be acquired by experimentation.

We are presently developing a flight management trainer which will have the functions described above. We are using PLATO author language for implementing various pedagogic situations. AIRBUS trainees are the subjects for the incremental knowledge acquisition. At this stage, a first ITS mockup has been developed an an IBM PC. This includes a technical KB and a sub-optimal pedagogic KB. The observer module is being developed using NEXPERT Object.

Acknowledgements:

This work was supported by AIRBUS INDUSTRIE contract n° 304 020.

Special thanks to P. Medous, F. Hernando and AEROFORMATION intructors for their excellent comments and advice. Thanks to AIRBUS INDUSTRIE test pilots and FMS experts for providing us detailed knowledge on FMS. Thanks to Philippa Gander and Alain Garès for critical comments on the manuscript.

References


