Explicit Reflection in Prolog-Tutor

Josephine Tchetagni, Roger Nkambou, Jacqueline Bourdeau

Explicit Reflection in Prolog-Tutor

Josephine Tchetagni, Roger Nkambou, *GDAC, Computer Science Department, UQÀM, Montréal, Québec H3C 3P8 Canada*
josephine.tchetagni@licef.teluq.uqam.ca, *nkambou.roger@uqam.ca*

Jacqueline Bourdeau, *LICEF Research Center, Télé-Université, Montréal, Québec H2T 3E4 Canada*
bourdeau@licef.teluq.uquebec.ca

Abstract. This paper describes a reflection-based approach for open learner modeling (OLM). Tutoring dialogues are used by learners to explicitly reveal their own knowledge state to themselves. Dewey's theory of reflective thinking is used to create tutorial strategies which govern these dialogues. Drake's specification of critical thinking, associated to a defined set of skills, is used to define tutoring tactics implementing these strategies. The main contribution of this approach to OLM is that it provides a set of principled and reusable tutorial strategies and tactics to promote reflection, as they are based on domain independent theories. Furthermore, an evaluation of such a principled approach to OLM is straightforward in certain cases, as it refers to theories which already provide evaluation criteria. The approach is integrated in Prolog-Tutor, an existing intelligent tutoring system for Logic Programming. This paper presents a qualitative study of the resulting system, based on think-aloud protocols. A result analysis reveals that explicitly fostering reflection supports reflection based OLM and provides landmarks to explain its manifestations. However, the results also suggest that this openness may be less helpful when used by learners who have already honed a high level of proficiency in logic programming.

Keywords. Reflection, open learner modeling, tutoring dialogues

INTRODUCTION

Metacognition enforces learning within a domain, as well as knowledge transfer across domains. Open learner modeling (OLM) is a recent research trend in learner modeling. OLM advocates the stimulation of metacognition by allowing learners to access, consult and interact with their own models (Cumming & Self, 1991). Metacognition is sometimes defined as the "awareness of one's own cognitive processes and cognitive state" (Flavell, 1979). Thus, reflective thinking can be considered the foundation of a metacognitive activity as participants observe their own learning process. Schön (1983) describes two types of reflections: reflection-in-action and reflection-on-action. Open learners' models are mostly used as tools to elicit reflection-on-action. Learners can examine their mastery level of the skills pertaining to a field of study or their level of knowledge of a field of study (Bull, McEvoy, & Reid, 2003; Zapata-Rivera & Greer, 2003); they can view peers' models in order to support collaborative learning, reflect by viewing (Vassileva, McCalla, & Greer, 2003) and edit their model in order to modify or negotiate its content (Bull & Pain, 1995).

The current conceptions of open learners' models do not explicitly promote reflection-in-action. Reflection-in-action prepares the learner to capture the contents of a learning domain, the reasoning strategies specific to that domain, the tutors' expectations, etc. Most importantly, reflection-in-action

introduces learners to their own cognitive state. Tutoring dialogues are generally the preferred tactics to support open learner modeling for reflection-in-action in ITSs. However, researchers have often outlined the fact that the reflective activities that emerge from tutoring dialogues are a side effect of the interactive nature of these dialogues (Dimitrova, 2003). The question which arises from that remark is that these interactive dialogues should embed goals which *explicitly* aim at promoting students' reflection. One must ensure that learners effectively mirror their mental state pertaining to a field of study and tutoring dialogues could embed goals which *explicitly* aim to foster students' reflection. Two main properties are desirable in such dialogues: (1) their structure should warrant a coherent and continuous focus towards a learning goal and (2) their contents (the communicative acts) should provide *concrete evidence* that, besides leading learners to construct a successful answer to a problem, they allow them to reflect on the targeted skills. To achieve this, one approach consists of interpreting a theory of reflective thinking while modeling these tutoring dialogues. The challenge of defining reflection has been studied by several scholars over the years. Lewin's model of action research is oriented towards reflection during problem-solving in social and organizational settings (Lewin, 1948). This model would be inappropriate in the context of one-on-one tutoring dialogues in Intelligent Tutoring Systems (ITSs), since it intrinsically relies on the dynamics of social interactions. John Dewey also suggested a theoretical basis for reflection in an educational context (Dewey, 1933). Dewey's pioneering work on a theory of reflection relies on logical and philosophical arguments to explicitly articulate the components of reflection. This theory has been widely recognized and it served as an inspiration for most contemporary models of reflection in experiential learning (Kolb, 1984) and professional education (Schön, 1983).

This paper follows from an earlier work on an approach to support explicit reflective thinking in Prolog-Tutor, an Intelligent Tutoring System (ITS) for logic programming (Tchetagni, Nkambou, & Bourdeau, 2005). At first, Prolog-Tutor supported OLM by fostering reflection-in-action implicitly, using a tutoring dialogue. The purpose of this paper is to introduce explicit reflection in Prolog-Tutor using Dewey's theory of reflective thinking. The advantage of such an approach is that the conception of tutorial strategies and tactics which promote reflection is based on formal principles. Henceforth, these strategies and tactics can be formally justified, reused and assessed using those principles. This new version of Prolog-Tutor, enhanced by introducing explicit reflection will be referred to as *ER-Prolog-Tutor*.

The remainder of the paper is organized into four sections. The next section describes the main features of Prolog-Tutor, outlining how the system enables OLM by fostering reflection implicitly. The following section presents ER-Prolog-Tutor, the result of integrating explicit reflection in Prolog-Tutor as a principled mean to promote reflection based OLM. After that, there is a section which describes the system architecture and implementation which underlie the features of Prolog-Tutor and of ER-Prolog-Tutor. The next section summarizes the results of a qualitative study of explicit reflection in ER-Prolog-Tutor. The goal of the study is to analyze the extent to which ER-Prolog-Tutor can actually trigger learners' reflections as explained by Dewey. Results from tape-recorded participants using think-aloud protocol are analyzed and they suggest explanatory hypotheses about four main points: (1) the characterization of how reflection is manifested when learners interact with ER-Prolog-Tutor's tutoring dialogues and how they access their own mental state when doing so; (2) the awareness of explicit reflection when learners interact with such dialogues; (3) the arising reflection patterns that deviate from Dewey's theory when using those ER-Prolog-Tutor tutoring dialogues; (4) the impact of ER-Prolog-Tutor system implementation on the benefits of its use to stimulate reflection. The final section of the paper outlines the lessons drawn from the study.

GENERAL DESCRIPTION OF PROLOG-TUTOR

Prolog-Tutor is an ITS intended to support learning of basic notions in logic programming. Prolog allows formalizing declarative knowledge and implementing reasoning processes about this declarative knowledge. Learning basic notions in Prolog generally bears on: (1) learning the vocabulary of the data structures used to represent information (e.g. how is declarative knowledge expressed in Prolog?) and (2) learning the principles governing the reasoning algorithms on those data structures.

Prolog integrates four basic data structures[1]: `variables`, `constants`, `compound terms`, `Horn clauses`. Higher level concepts could be associated to these data structures: `facts` (based on `compound terms`), `Prolog rules` (based on `Horn clauses`), `knowledge bases` (based on `facts` and `Prolog rules`) and `goals` (as `facts` to prove using a `knowledge base`). Two main algorithms support the simulation of reasoning in Prolog: `unification` and `resolution`: `unification` allows the verification of a `fact` in a Prolog `knowledge base` and `resolution` implements logical proofs using `unification` to this end.

Different abilities may be applied to these data structures and algorithms. For example, to acquire `resolution` as a concept (or to "*understand* how `resolution` is used to prove `a goal`"), it suffices to "*understand* `unification`". However, to acquire `resolution` as a procedure (or to "*perform* the proof of `a goal` using `resolution`"), one must "*understand* how `resolution` is used to prove `a goal`" and to "*perform/apply* `unification`". Taking this into account, these data structures and algorithms are considered as knowledge elements. In the framework of this paper, associating a knowledge element with an ability defines a logic programming skill (Nkambou, Frasson, & Gauthier, 2003). The set of skills represented in Prolog-Tutor includes the skills related to Prolog vocabulary and those related to the `unification` and `resolution` algorithms. Table 1 presents the skills which are related to proving `a goal` using `resolution` in Prolog (also called "`resolve a goal`" in Table 1).

In order to apply the `resolution` algorithm, a learner should be able to: *interpret* a `fact` as a `goal` to be proven, *use* a `Prolog rule` to solve a goal, *manipulate* `a Prolog rule` to prove a `goal`, etc. Notice that the skills are defined here using action verbs which apply to a knowledge element in the field of study. The main advantage of defining a skill this way is that as abilities are generic processes, pedagogical strategies and tactics to support reflection on a skill could be designed based on these abilities, regardless of the corresponding knowledge element. Thus, as domain contents are independent, these strategies and tactics are reusable across domains.

Prolog-Tutor pedagogical actions focus on enabling OLM by fostering reflection on skills diagnosed as being the cause of learners' shortcomings. This section introduces the basic components of Prolog-Tutor, the pedagogical model being emphasized as it sustains OLM in this system.

---

[1] For clarity, the font "Courier New" depicts expressions that are proper to Prolog Language or to PROLOG-TUTOR

Table 1
Skills related to the "resolution of a goal" in Prolog

Prolog Skill	Knowledge Element	Skill According to Bloom's and Gagne's Taxonomies
Interpret a (Fact) as a goal to be proven	Fact	Understand Propositions
Use a (Prolog-Rule) to resolve a goal	Prolog Rule	Apply Principles
Manipulate a (Prolog Rule) to resolve a goal	Prolog Rule	Apply Principles Apply Procedures
Use a (Knowledge base) to resolve a goal	Knowledge Base	Apply Principles
Understand (Back-Tracking) to resolve a goal	Back-Tracking	Understand Concepts
Use (Back-Tracking) to resolve a goal	Back-Tracking	Apply Principles
Execute (Back-Tracking) to resolve a goal	Back-Tracking	Apply Principles Apply Procedures

Prolog-Tutor Domain Model

The choice of the knowledge representation approach of the learning domain was motivated by the cognitive diagnosis procedures applied by the system. Cognitive diagnosis in ITS provides various types of information: learners' knowledge state, causes of learners' errors and forecasting of learners' future actions in problem solving. In Prolog-Tutor, these inferences are supported by a causal model defining causal relationships among the skills represented in the learning domain (Figure 1). Moreover, diagnosis reasoning binds observations and explanations together. A number of explanations may apply to a single observation and consequently require the identification of the most plausible explanations. Bayesian networks are appropriate knowledge representation approaches to support diagnosis reasoning: they allow representations of causal relations and account for the uncertainty when reasoning about such relations (Pearl, 1988).

Creating the domain model was a two-step process. First, causal relations between the considered skills were identified. Two lecturers in logic programming in the Department of Computer Sciences at the University of Quebec in Montreal performed this task. Prolog can be considered a well structured domain as, quite often, more complex skills are built from simpler skills. As presented in this paper, Prolog-Tutor focuses on the topic pertaining to prove a goal using resolution. A goal proof is built on the application of the resolution procedure, which itself is built upon simpler skills such as "apply unification of two compound terms", "understand unification as a concept", "use a knowledge base", "understand the concept of bound variable", etc. A total of twenty-six (26) such skills were identified and integrated into the domain model; among these skills, fifty-seven relations were identified with a maximal number of three causal relations between two skills. Second, these causal relations were

quantified by associating conditional probabilities to each of them, using an expert centric approach (Mayo & Mitrovic, 2001). These probabilities were established using the intuition and experience of both lecturers. Figure 2, for example, presents a portion of the Prolog-Tutor domain model. There is a causal link between the skills "Apply the Unification of two compound terms", the "Use a Bound Variable", "Apply Unification" and "Articulate the structure of a Compound Term".

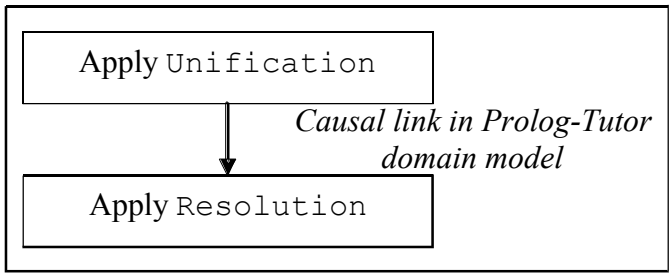


Fig.1. Example of Causal Link between Two Skills in Prolog.

Most of the time required to build the domain model was spent in this second phase. The posterior probabilities of each node had to be set for all the possible values of its parents, which was rather challenging in the presence of more than two parent nodes. The time required for assigning these probabilities was not tracked. However, the process was long enough to reveal some shortcuts to resolve in subsequent versions of Prolog-Tutor. First, the probabilities were assigned subjectively thus, their validation remains an issue. Second, the fact that the probabilities were encoded manually questions the scalability of the Bayesian approach: what would happen if more skills were incorporated in the system? These two issues could be addressed by using a data centric approach to build the Bayesian network (Mayo & Mitrovic, 2001). In that case, both the structure and conditional probabilities of the network are gathered from data collected from real-world evaluations of the tutor or of a human tutor. In the case of this investigation, the corpus of data necessary to generate these probabilities was not available. The logic programming topic selected was very specific (`proof of a goal` using `resolution`) and did not require a large number of skills so the hard-coding solution was a fair compromise.

Prolog-Tutor Learner's Model

Two types of diagnosis inference are expected in Prolog-Tutor: update learners' knowledge state (what is the mastery level of a given skill?) and explain learners' errors (why does a learner experience difficulties with specific elements?). Updating the learner's knowledge state consists of establishing the probability of mastering the skills represented in the domain model. Diagnosing learner's errors consists of finding the most probable explanations for these errors (for instance, the lack of a certain skill). When using Bayesian networks, these two types of inferences are automatically handled by the *Belief updating* and *Belief revision* algorithms (Pearl, 1988).

Diagnosing the Learners' Knowledge State

When updating the learners' knowledge state, the mastery of a skill can be represented by a binary random variable: a value of 1 means that the skill was successfully mastered while a value of 0 indicates the contrary. This part of the learner's model represents the probability that each skill is mastered (state of mastery equals 1). Diagnosing the learner's knowledge state consists of updating the probabilities that each skill is mastered, based on the learner's actions. For example, every time learners successfully complete an exercise or correctly respond to a question, *positive evidence* is registered by the Bayesian network associated with the domain model. Every time evidence is registered, it is propagated throughout the network to allow computing future probabilities of all the nodes in that network. Figure 2 illustrates such propagation.

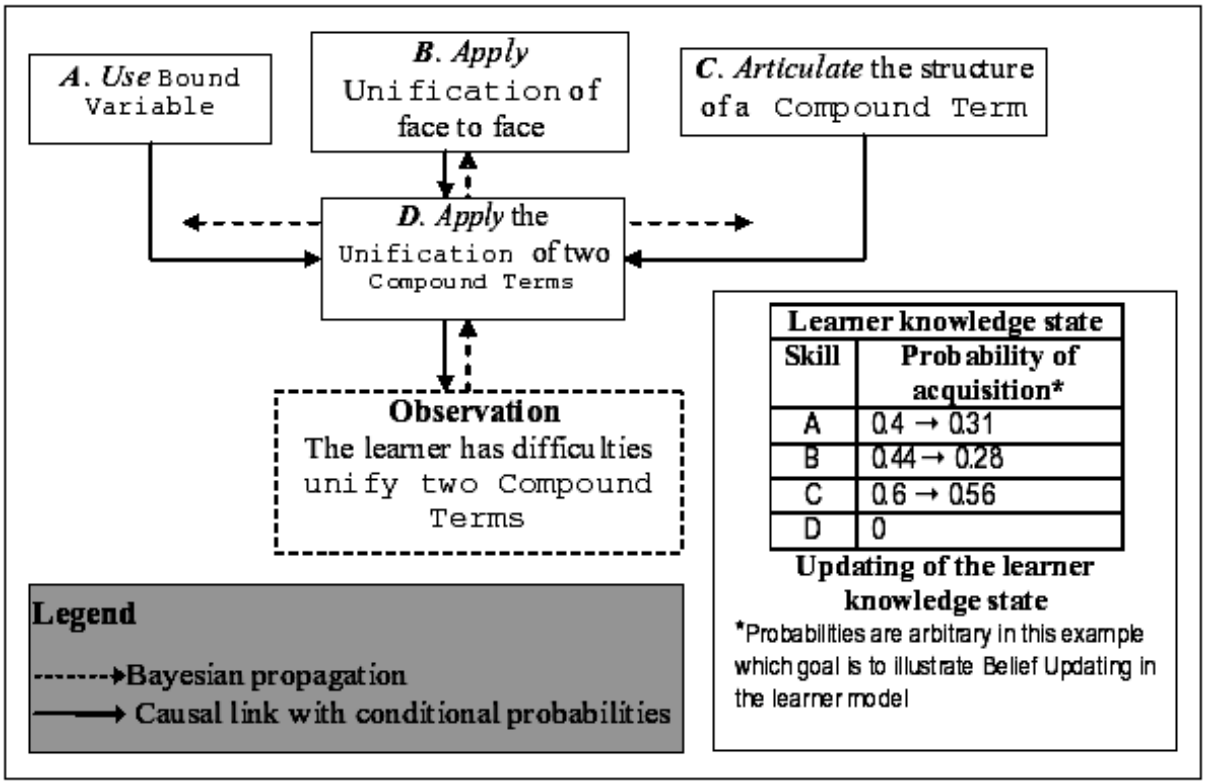


Fig.2. Updating a Learner's Model with Bayesian Deduction.

The learner fails to *apply* the `Unification of two Compound Terms` and this stands as negative evidence related to the skill "*Apply* `Unification` of two `Compound Terms`". This evidence is propagated through the network and new probabilities are computed. For example, the posterior probability of mastering the skill "*Use* a `Bound Variable`" decreases from 0.4 to 0.31, in the situation illustrated in Figure 2.

Diagnosing Learners' Difficulties

Diagnosing learners' difficulties occurs when learners fail to answer a question while interacting with the system. Each question in Prolog-Tutor is related to a specific skill or to a set of skills. First, skills are diagnosed. Whenever learners do not provide one of the expected answers, negative evidence is registered in the Bayesian network: the values of the nodes representing the diagnosed skills are set to 0 (skill not mastered). The next step consists of finding an explanation for that observation: find the network state with the highest joint probability, given that the values of the nodes associated to the diagnosed skills are set to 0. In Figure 3 for example, the goal is to find the state of the network which best explains the fact that the node "*Apply* `Unification` of two `Compound Terms`" is associated with a 0 value (skill not acquired). Three network states are simulated with the value of the node "*Apply the* `Unification` of the arguments of two `Compound Terms`" (node labeled as D) fixed at 0. For example, the first network state includes the node values A = 1, B = 1, C = 1, D = 0 and its joint probability equals 0.0054. The network state with the greatest joint probability is represented by the node values A = 0; B = 0; C = 1; D = 0. Thus, the lack of the "*Use a* `Bound Variable`" and "*Apply the* `Unification` of the arguments of two `Compound Terms`" skills best explains the fact that the learner has difficulties `unifying` two `Compound Terms`. The next section describes how Prolog-tutor makes use of tutoring dialogues to diagnose learners' knowledge state and errors.

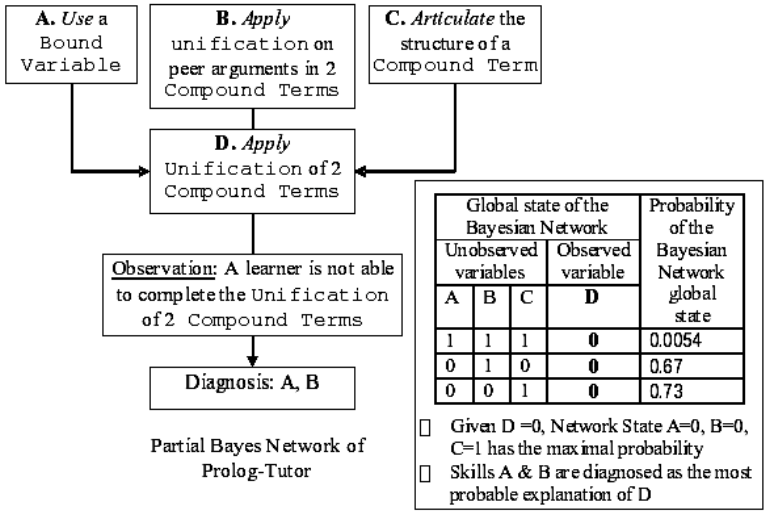


Fig.3. Bayesian Induction to Diagnose Learners' Errors.

Using Tutorial Dialogues for Learner Modeling

Learners interact with Prolog-Tutor in two ways. First, they can consult general contents pertaining to a skill (e.g., definitions, examples, explanations). Second, they can complete exercises where they must elicit logic programming skills. Learner modeling builds upon evidence collected in exercise-based training. The construction of the solutions to these exercises is incremental, based on tutoring dialogues. These dialogues include questions intended to diagnose logic programming skills, hence

learners' answers can be used as evidence of their level of acquisition. Figure 4 presents an English translation of the typical interface for a tutoring dialogue in Prolog-Tutor. The exercise statement lies in the upper left-hand corner of the interface. In this case, learners must *apply a* `resolution` to prove a `goal`: they must find the values of `variables` X and I so that the `fact` `x_has_illness(X,I)` be `true`, given a Prolog `knowledge base`. This `knowledge base` is defined in the upper right-hand corner of the interface. The tutoring dialogue appears on the bottom part of the interface. The "Implementation" section of this paper presents a detailed description of the Prolog-Tutor interface commands.

The tutoring dialogue is composed of a two-level hierarchical structure. On the first level (called D1 in the remainder of the text), each question corresponds to a sub-problem associated with the current exercise. This sub-problem is explicitly associated with a skill that must be elicited in order to provide a successful solution. Therefore, these questions are considered diagnostic questions (DQ). The learner's answer to a DQ determines how the dialogue evolves: if the response is correct, the next DQ is addressed; otherwise, the tutor focuses the interaction on the skill associated to that DQ in order to determine which aspect of that skill is problematic for the learner.

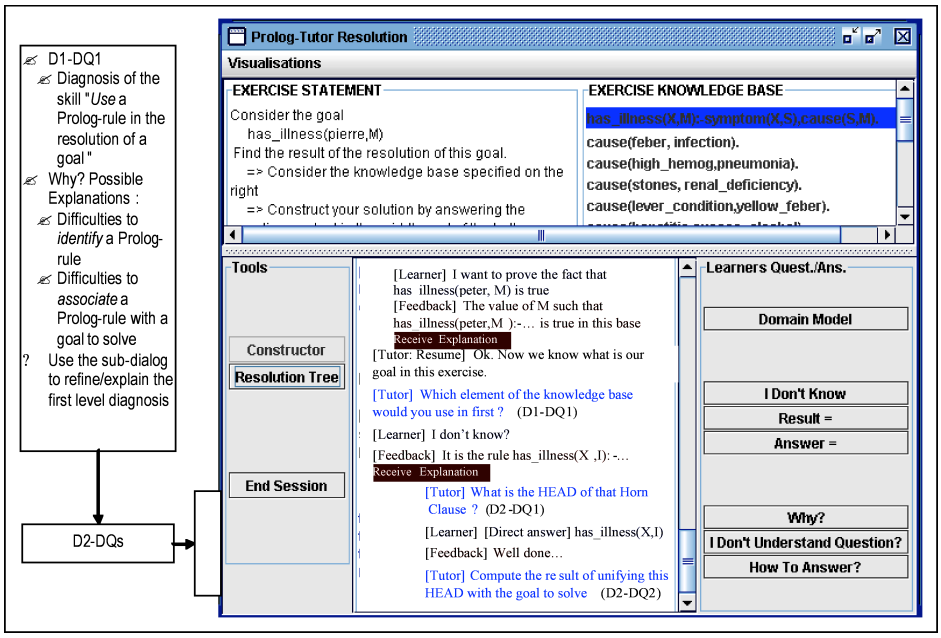


Fig.4. Learner Modeling in Prolog-Tutor Using a Tutoring Dialogue.

Learners who fail to answer D1 Level DQs correctly indicate their lack of the skill associated to that DQ or that they fail to understand the DQ. Indeed, a learner can signal that a question is not understood by clicking on a "Don't Understand" button provided on the user interface of the tutoring dialogue (Figure 4). In cases where a diagnosis is made with respect to a skill, the tutoring strategy consists of focusing the learning process on that skill, rather than continue constructing the problem solution. If a diagnosed skill contains sub-skills, the second level of the hierarchical structure is triggered as a sub-dialogue. The sub-dialogue of the D1 dialogue level is called D2. Three tactics can be considered to determine the sub-skills that need to be focused on at Level D2. The sequential tactic

consists of focusing on all the sub-skills sequentially. The adaptive tactic consists of focusing on sub-skills for which the probability of mastery is below a given threshold. The "most-probable-explanation" (MPE) tactic uses Pearl's Belief revision algorithm to identify the lacking sub-skills which best explains the absence of the diagnosed skill (Tchetagni & Nkambou, 2004). In each case, the questions asked at level D2 correspond to the sub-skills that are chosen to be focused on. The choice of any such strategy must be set manually by the author before a learning session begins.

Example of a Diagnosis Based on the Tutoring Dialogue

In Figure 4, the tutor asks the learner a question related to (the skill) *Use* a `Prolog-Rule` while applying `resolution` to prove a `goal` (question labeled D1-DQ1 in the illustration): [*what is the first element of the prolog* `knowledge base` *which corresponds to that* `goal?`]. If the learner fails to answer, the system: (1) diagnoses that the corresponding skill may be lacking and (2) generates a sub-dialogue in order to provide the learner with an insight into this skill through an articulation of the sub-skills that are most probably responsible for the learner's obvious difficulties. In this case, the sub-dialogue addresses two skills (with the questions respectively labeled D2-DQ1 and D2-DQ2 in the illustration): *Identifying* a `Prolog-Rule` and *Use* a `Prolog-rule` with a `goal` to solve.

Here, the dialogue structure includes two pedagogical goals. First, each DQ in D1 allows the system to diagnose learners' difficulties directly. Second, each DQ in D2 gives the learner the opportunity to reflect implicitly on the skills identified as lacking at the preceding level. This nesting of dialogue levels is relevant because the system performs an interactive diagnosis which fosters an implicit reflection and each question at Level D2 is also associated with a skill so that the cognitive diagnosis is continuously refined, confirmed or invalidated.

Prolog-Tutor Pedagogic Model: Openness of the Learner Model

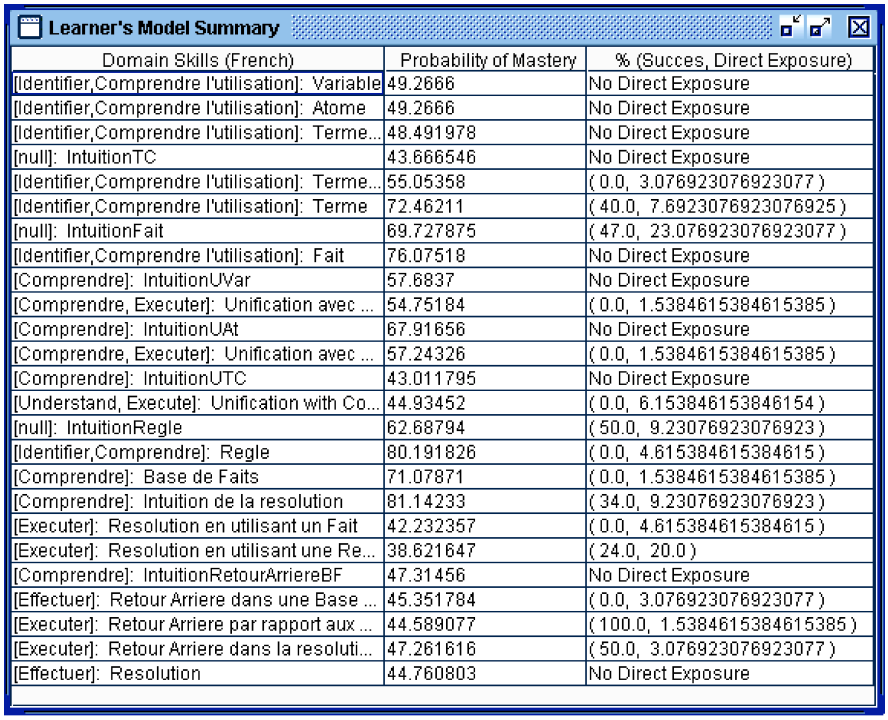
The pedagogical module of Prolog-Tutor focuses on stimulating metacognition on skills diagnosed during learner modeling. To support the learner's metacognition, two approaches are used to access the learner model. In the first approach (OLM based on reflection-on-action), learners can view and interact with a representation of their cognitive states as interpreted by the system. In the second approach (OLM based on reflection-in-action), tutoring dialogues are used to sensitize learners to their own cognitive state in a given learning situation in Prolog-Tutor. Indeed, this approach to OLM can be used by learners to prepare subsequent interactions with the representation of their cognitive state by the system.

OLM in Prolog-Tutor through Reflection-on-Action

Learners can consult and modify two main components when consulting their model: a representation of the long term knowledge state and a learning trace indicating relevant information about their last training session.

Figure 5 presents an example of a representation of a learner's long term knowledge state in Prolog-Tutor. For each logic programming skill represented in the domain model, the long term knowledge state indicates the following: its probability of mastery (computed by the Bayesian network), the learners' score (in percentage of success) for exercises which require that skill and the percentage of times that the learner performed an exercise where that skill intervenes. Such data is

intended to provide more insight about a given probability. It provides cues about the number of times that this probability is computed by the system, based either on direct evidence from the learner or on Bayesian inferences. The left-hand side column indicates the list of skills contained in the domain model. The column in the middle indicates the probability of mastery of each skill and the right-hand side column summarizes the diagnostic sources of the column in the middle using the pair *(success percentage, direct exposure)*.



Learner's Model Summary

Domain Skills (French)	Probability of Mastery	% (Succes, Direct Exposure)
[Identifier,Comprendre l'utilisation]: Variable	49.2666	No Direct Exposure
[Identifier,Comprendre l'utilisation]: Atome	49.2666	No Direct Exposure
[Identifier,Comprendre l'utilisation]: Terme...	48.491978	No Direct Exposure
[null]: IntuitionTC	43.666546	No Direct Exposure
[Identifier,Comprendre l'utilisation]: Terme...	55.05358	(0.0, 3.076923076923077)
[Identifier,Comprendre l'utilisation]: Terme	72.46211	(40.0, 7.6923076923076925)
[null]: IntuitionFait	69.727875	(47.0, 23.076923076923077)
[Identifier,Comprendre l'utilisation]: Fait	76.07518	No Direct Exposure
[Comprendre]: IntuitionUVar	57.6837	No Direct Exposure
[Comprendre, Executer]: Unification avec ...	54.75184	(0.0, 1.5384615384615385)
[Comprendre]: IntuitionUAt	67.91656	No Direct Exposure
[Comprendre, Executer]: Unification avec ...	57.24326	(0.0, 1.5384615384615385)
[Comprendre]: IntuitionUTC	43.011795	No Direct Exposure
[Understand, Execute]: Unification with Co...	44.93452	(0.0, 6.153846153846154)
[null]: IntuitionRegle	62.68794	(50.0, 9.23076923076923)
[Identifier,Comprendre]: Regle	80.191826	(0.0, 4.615384615384615)
[Comprendre]: Base de Faits	71.07871	(0.0, 1.5384615384615385)
[Comprendre]: Intuition de la resolution	81.14233	(34.0, 9.23076923076923)
[Executer]: Resolution en utilisant un Fait	42.232357	(0.0, 4.615384615384615)
[Executer]: Resolution en utilisant une Re...	38.621647	(24.0, 20.0)
[Comprendre]: IntuitionRetourArriereBF	47.31456	No Direct Exposure
[Effectuer]: Retour Arriere dans une Base ...	45.351784	(0.0, 3.076923076923077)
[Executer]: Retour Arriere par rapport aux ...	44.589077	(100.0, 1.5384615384615385)
[Executer]: Retour Arriere dans la resoluti...	47.261616	(50.0, 3.076923076923077)
[Effectuer]: Resolution	44.760803	No Direct Exposure

Each part of a learner's model could be opened separately. Above, a summary of the learner's knowledge state is presented in three columns with (left to right): skills; probability of mastery; percentage of success for each skill

Fig.5. OLM in Prolog-Tutor: Consulting a Learner's Long Term Knowledge State.

For a given learner, two kinds of information are included in a learning trace: the posterior probability of mastering each skill, based solely on evidence from previous training sessions (regardless of the long term probability of mastery) and a representation of all of the learner's actions during the last training session. Figure 6 presents an interface which illustrates the learning trace. The left-hand side lists the identifiers of the skills represented in the Prolog-Tutor domain model. Each skill identifier is preceded by a coloured square indicating information pertaining to the diagnosis of the corresponding skill in the last learning session: a red square means that a learner provided an erroneous response, a green square shows that the learner successfully answered the question and a grey square indicates the absence of evidence. The right-hand section of the interface comprises two horizontal panels. The upper panel reports the contents of the tutoring dialogue during the last training session, represented as numbered questions and answers. Each tutor question is also labeled with the

name of the underlying tutoring strategy in order to stimulate reflection. The bottom panel provides learners with tools that allow them to influence the contents of their model. When a skill is selected on the left-hand side of the interface, this bottom panel presents the long and short term probabilities of mastery for that skill and the source of the last diagnosis whose result included that skill (the source is identified by the interaction learning trace number where the diagnosis was performed). For example, in Figure 6 the short term probability of mastering the skill "*Apply* `Unification` of two `Compound Terms`" is 0 since the learner failed to answer the last question which required that skill. Notice that for this learner, the long term probability of acquiring that skill equals 0.1748. In order to involve them in their diagnosis, learners are allowed to influence their model using the interface of the short term learner's model. For a given skill, they may eventually judge that their mastery level is superior to that indicated (In Figure 6 they would click on the "I do not agree" button). The use of this button triggers the launch of a control exercise requiring that skill which learners must take in order to justify their claim. If the control exercise is successfully completed, the tutoring system updates the learner's model accordingly. The data on the bottom panel could also be useful for human tutors to detect phenomena such as guessing (when there is a large difference between the long and short term probabilities of mastering a skill).

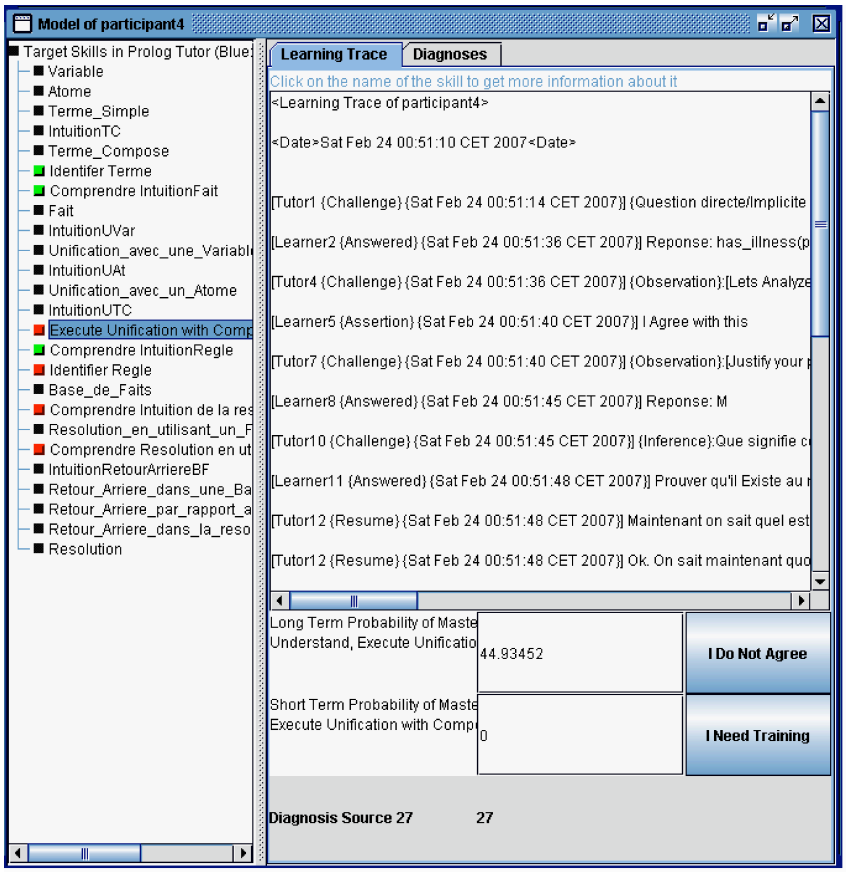


Fig.6. OLM in Prolog-Tutor using the Learning Trace.

Giving learners the opportunity to analyze their cognitive state as interpreted by the system is pedagogically valuable as it may provide insights into their understanding of the learning domain through reflection-on-action. This paper suggests that learners' prior preparation to this analysis could improve performance. One way to prepare learners for consulting their model consists of promoting an insight into their understanding of the domain at the onset of their training in the system, through reflection-in-action.

OLM in Prolog-Tutor through Implicit Reflection-in-Action

In Prolog-Tutor dialogues, each question should be sufficiently suggestive so that it fosters a hidden (implicit) reflection which could then remedy the cause of the learner's error: re-examine data pertaining to the current state of the problem, recall a principle pertaining to the domain, apply a principle in order to produce an answer. In the example, depicted in Figure 4, Level D1-DQ: "*What is the first element of the knowledgebase that should be used to solve the goal has_illness(Peter,I)*" is suggestive as it may hint to the learner that the knowledge base should be scanned sequentially according to the syntactical form of that goal. In order to understand the question, the learner must reflect on the principle which is required to elicit the skill "*use* a Prolog-Rule in the proof of a goal based on resolution":

> [IF the HEAD of a PROLOG-RULE is UNIFIABLE with the GOAL to prove, THEN *use* that PROLOG-RULE to perform the RESOLUTION]

The sub-dialogue illustrated in Figure 4 tackles reflection on the first sub-skill of the skill [*Use* a Prolog-rule in the proof of a goal based on resolution], namely the [*Identify* a Prolog-rule] skill. The purpose is to create a situation where learners must become aware of the concept of "prolog-rule" in order to consider their understanding of that concept by themselves. In this example, the learner has to *identify* a prolog-rule in the context of the proof of a goal. The tutor asks the learner to indicate the "Head" of the prolog-rule that should be used indeed, to prove the current goal.

It is likely that a reflective process implicitly takes place through the use of this tutorial dialogue. However, from a pedagogical perspective, this approach does not ensure that the learner truly elicits the targeted skills. The next section describes ER-Prolog-Tutor, a version of Prolog-Tutor where reflection-in-action based OLM is supported using Dewey's theory of reflective thinking to explicitly promote that reflection.

OLM IN ER-PROLOG-TUTOR THROUGH EXPLICIT REFLECTION

In the previous description of Prolog-Tutor, even though the presence of reflective thinking relies on the interrogative nature of the interactions, there is no guarantee that reflection actually occurs. Our goal is to design the sub-dialogues of the Prolog-Tutor tutoring dialogue structure (at Level D2) so that they explicitly trigger reflection. When tutorial strategies are not based on a specific learning or teaching model, they are difficult to justify, select, reflect upon and improve (Ford, 1987). Our objective is to derive generic and reusable tutoring strategies and tactics from the principles of Dewey's theory and to implement them in ER-Prolog-Tutor, the Prolog-Tutor version where tutoring dialogues explicitly foster learners' reflection. A tutoring strategy defines a tutor's goal or sub-goal

towards a learning state, given a learner's current knowledge state; a tutorial tactic defines the means by which a tutoring strategy is to be achieved: the surface manifestation of a strategy in terms of tutoring actions (Ohlsson, 1987). The tutorial strategy in ER-Prolog-Tutor is designed by defining the goal of each question in the tutoring sub-dialogues at Level D2 in order to elicit a component of reflective thinking. The corresponding tactics are based on the idea that the contents of these dialogues refer to the nature of the skill on which the learner reflects (the skill diagnosed at Level D1), as defined in Gagne's taxonomy of skills (Gagne, 1992).

Pedagogical Strategies for Explicit Reflection: Dewey's Theory

According to Dewey (Dewey, 1933), reflective thinking refers to the occurrence of neither ideas nor judgments. Reflection pertains to the way in which a belief was used to construct a conclusion. A conclusion is drawn *on the grounds of reasoning* and a set of *evidence* which supports that reasoning. Dewey identified five components of reflective thinking. The first component consists of a contradictory/controversial/obscure experience of a situation, where learners must reason in order to draw out an acceptable conclusion/explanation. The second component refers to an intellectualization process triggered by the controversial situation/experience: it implies a formal definition/identification of the conditions (the facts) which cause the contradiction/controversy/obscurity. The third component relates to the process of proper reflection: it consists of pondering upon the facts of a situation in order to reach an acceptable conclusion. The fourth component is facultative and refers to internal reflection: the reflective learner can mentally process related ideas in order to infer further hypotheses. Finally, the fifth component is the result of reasoning, which consists of selecting a single conclusion amongst those applicable to solve the situation presented at the first step. This conclusion could eventually be tested if it does not consistently follow from the facts which characterize the situation.

The first component of reflective thinking requires that learners' curiosity be stimulated in a learning situation. The second component emphasizes the importance of explicit analysis of facts in a learning situation, as in a scientific experimentation. The third component integrates the learning situation and the learning domain: conclusions must be built by reflecting upon the basis of the facts of that situation, using the domain knowledge. The fourth component is similar to the third since it concerns reasoning, but it also allows the introduction of the effect of experience in the reasoning. The fifth component allows the proper completion of the reflection process by verifying the drawn conclusion. Four tutoring strategies are suggested to encompass these goals: (1) Elicit curiosity, (2) Identify the relevant facts and conditions of the learning situation contents, (3) Draw a solution from the learning situation by linking the identified facts to some principles, concepts, heuristics or past experience in the learning domain (this strategy could be reified at any step of the reflection), (4) Evaluate the correctness of the solution, given the facts and the domain principles identified.

Pedagogical Tactics to Enable Explicit Reflection in Prolog-Tutor

In order to enable the tutorial strategies for explicit reflection defined above, we propose to design the sub-dialogues of ER-Prolog-Tutor (Level D2) so that they explicitly trigger reflection on the skill associated to the corresponding diagnosis question at the upper level (Level D1). When the system diagnoses that a learner lacks knowledge about a particular skill (indicated by diagnosis questions used at Level D1, the first hierarchical level of the dialogue), its goal is to elicit an explicit reflection

pertaining to the acquisition of that skill (using tutorial strategies for explicit reflection to guide the nature of the questions at Level D2, the second hierarchical level of the dialogue).

Consequently, the sub-dialogue is designed in such a way that it interprets Dewey's components of reflective thinking. However, this dialogue model should also achieve a certain degree of generality in interpreting Dewey's components on the basis of the nature of a skill rather than specifically for the Prolog domain. A skill can be sufficiently general (regardless of the context in which it is used) to be defined with respect to a skill taxonomy, which gives a principled description of its learning properties. The following interpretation method was selected: first, determine the goal of the reflective process regarding each type of skill and second, infer how (to interpret) the components of reflective thinking can be used to reach these goals. Concerning the goal of reflection on a particular type of skill, Dewey outlined that the intellectual activities which underlie the process of judgment are analogous to reflective thinking. Therefore, Drake's (1976) description of the goals of judgment is relevant, since these goals are defined specifically for concepts, principles and arguments as presented in Table 2.

Table 2
Skills and Objectives of Reflective Thinking (Drake, 1976)

Skill	Objective of Reflective Thinking
- Identify a concept	Determine if a concept instance classification is adequate; verify the meaning; recognize the attribute; justify an erroneous classification.
-Apply, Use Principles	Verify statement confirms/contradicts; identify premises /consequences.
-Verify Arguments	End up with contradiction/confirmation; determine assumptions/ consequences.

Then, a dialogue model for each targeted skill must be designed as a tactic to enable a strategy of reflection. The interpretation of the reflective process uses the characteristics of a skill as defined in a taxonomy. Using Gagne's taxonomy, a model of explicit reflective thinking about a principle is presented in Table 3.

Table 3
A Generic Tactic for Explicit Reflection on a *Principle*

Reflect on the Application of Principle P1 Goal of the Reflection: Determine whether a statement confirms or contradicts P1	
Steps	**Interpretation in the Context of 'Applying a Principle'**
Problematic Situation and Intellectualization	1- Describe several facts to which the principle could be applied. 2- Outline the possible facts which could be drawn from the situation.
Reasoning on the Principle	1- Make a statement S which corresponds to the targeted conclusion. 2- Verify whether P1 contradicts or confirms S.
Evaluation (Optional)	1- If S is an acceptable conclusion, no need for evaluation. 2- If S is an unacceptable conclusion, show the contradiction to the learners by having them evaluate S with P1.

This model stems from the fact that a principle is invoked in two possible ways: by the presence of the conditions to which it applies or by the presence of the consequences that it implies. In ER-

Prolog-Tutor, this model is applied when a learner is unable to answer correctly a Level D1 question. At Level D2, the sub-dialogues articulate the components of Dewey's reflection according to the nature of the skill diagnosed at Level D1. Most of the skills considered in ER-Prolog-Tutor pertain to the knowledge, use and application of a principle. For this reason, the model depicted in Table 3 applies to most of the system's sub-dialogues.

Concerning the presentation of a problematic and confusing situation, learners are presented with an assertion (which is related to the skill diagnosed at Level D1) where they have to indicate whether they agree or disagree. In general, the right response requires the use and application of a principle. Concerning the intellectualization of a problematic situation, if agreeing with the assertion is correct, the learner is asked to identify the facts that generally correspond to the conditions and to the prerequisites of the principle to use and to apply in order to reach the conclusion (the agreement/disagreement with the previous assertion). As for the reflection reasoning components: if the learners' position is correct (agree or disagree with an assertion presenting a problematic situation), they are asked to provide intellectualized facts with their position; if the learners' position is incorrect, they are asked a series of questions designed to lead them to contradict themselves, given the facts that they have intellectualized and the principle which applies to that specific case. Let us consider Figure 7 for example. The learner has difficulties finding the first element of a `Prolog knowledge base` that is used for the `proof` of a `goal` based on `resolution`. In the following sub-dialogue, the system states that the `prolog-rule`

"`has_illness(X,I):-symptom(X,S)…`"

should be the first element to use in order to prove the given `goal` and learners must indicate whether they agree (presentation of a problematic situation where the learner is forced to take a position). Then, the system asks the learner to identify the `head` element of that `prolog-rule` (intellectualization of the fact necessary to use and apply the principle corresponding to the right answer). The system finally asks the learner to establish whether this `head` element can be `unified` with the `goal` being proven (reasoning on the foundation of the fact that has been intellectualized), using a specific interface to articulate `unification`. Notice that the fourth component of Dewey's reflection (evaluation) was not used in this implementation of ER-Prolog-Tutor. We believe that it is most appropriate for contexts with richer contents and greater quantities of alternative solutions.

Explicit reflection is introduced in ER-Prolog-Tutor using tutorial strategies and tactics from two formal theories. The strategies are based on the components found in Dewey's theory of reflection. The tactics implementing these strategies are built according to the epistemic nature of the skill pertaining to reflection, using Gagne's epistemology and referring to Drake's theory of critical thinking. The advantage of such an approach is that it is based on formal principles and this stands as the main contribution of this paper. Therefore, the tutoring strategies and tactics used to promote explicit reflection in ER-Prolog-Tutor are systematic and reusable: the tutoring strategies are generic by themselves since they follow from a theory; the tutoring tactics apply to well defined sets of skills, namely, Gagne's intellectual skills. Thus, provided that the contents of a domain are modeled in terms of such type of skills, those tactics could be applied. More precisely, Intelligent Tutoring Systems for domains for which contents are expressed in terms of concepts, principles and laws could import the tactics presented here (for example: computer programming, language learning – more precisely, grammar – physics whose contents can be easily expressed in those terms).

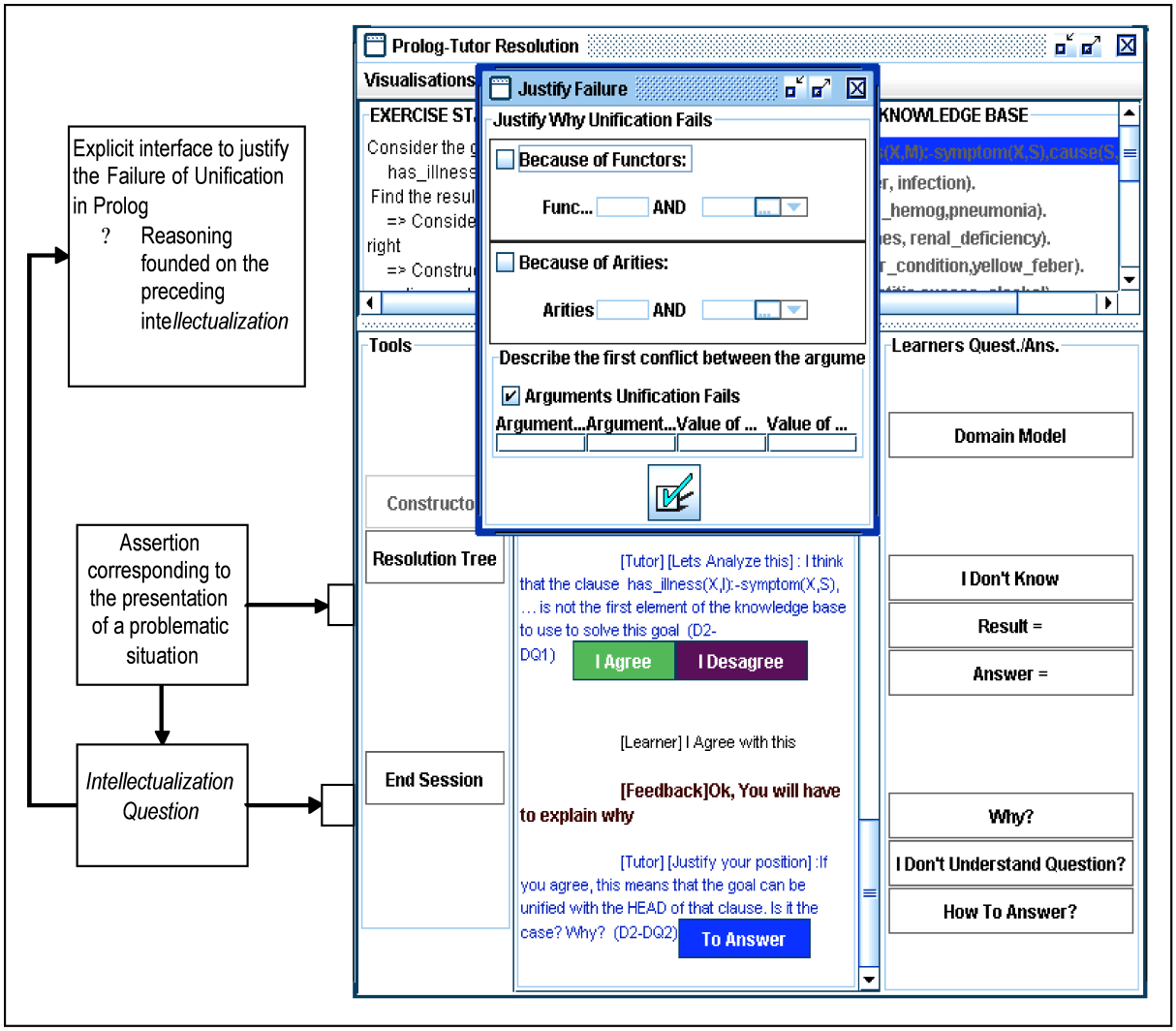


Fig.7. OLM in ER-Prolog-Tutor: a Tutoring Dialogue for Explicit Reflection.

IMPLEMENTATION

This section describes the implementation of the main components of Prolog-Tutor and ER-Prolog-Tutor: the domain, the learner, the communication and the pedagogical modules. Tutoring dialogues are part of the pedagogical model as tools to promote reflection and they also integrate the communication model as the medium of interactions evolving between a learner and the system. For the sake of simplicity, any description concerning Prolog-Tutor in this section also applies to ER-Prolog-Tutor, unless an explicit distinction is stated.

The Domain Module

The domain module comprises: a representation of the domain skills and the relations between each of them, a learning resource and a learning objective. A learning resource can be an exercise to complete or another instructional activity such as a request for definitions, examples or explanations. A learning objective is linked to a skill. XML-based metadata permitted the implementation of these representations. Causal relations among the skills and the corresponding conditional probabilities are materialized through an XML structure that is compatible with the data entry of the JavaBayes library algorithms from Cozman's Bayesian library (Cozman, 2001). Such algorithms were used to support the Bayesian inferences applied in the domain model to support learner modeling.

Learning Objectives

Learning objectives can be named, uniquely identified (with IDs) and associated with one or many learning activities. Learning activities are also named and uniquely identified and they are linked with one or many resources, which are described in separate XML files. They are uniquely identified, named and associated with a skill and linked with a Java class which represents its interface and dynamics. This representation of the relations between learning objectives, activities, resources and skills is inspired from the CREAM Model (Nkambou et al., 2003). This model permits a formal organization of learning sessions.

A learning objective is reached by executing one or many learning activities. Learning activities take place when learners use and interact with learning resources.

Learning Resources

ER-Prolog-Tutor comprises two types of learning resources: static and dynamic learning resources. Static resources materialize learning activities where learners grasp instructional content pertaining to the objective targeted by this activity; dynamic resources, also called "intelligent resources", materialize practical exercises where learners interact with the system in a more sustained manner in order to solve these exercises (using tutoring dialogues). At this time, ER-Prolog-Tutor implements one (1) type of exercise in Logic Programming, which carries on "`the proof of a goal using resolution`" based on a `Prolog knowledge base`. Two instances of such type are implemented: one instance pertaining to a `proof of a goal using resolution` and which results in a (Prolog) `Failure` and which does not involve any `back-tracking;` one instance pertaining to a `proof of a goal using resolution` and which results in a (Prolog) `Success` after `back tracking`.

Dynamic resources define the contents of the corresponding tutoring dialogue. These contents will be used by the communication and the pedagogical modules to conduct a tutorial dialogue. The representation of these resources in the domain module defines the contents of the corresponding dialogue as well as its encoding in the system.

1. Representation of the Dynamic Resources Contents

In dialogues, each tutor utterance consists of two components: a discourse marker and some contents (Moser & Moore, 1996)**.** The discourse markers signal the focus of the current tutor's utterance**:**

tutorial strategy underlying a dialogue question, tutor feedback, summary of a sub-dialogue. For example, each question representing the "intellectualization" phase of reflection as a tutorial strategy is preceded by the utterance: "*Now, I want you to examine the important data of this situation*". Prolog-Tutor includes four types of utterances: questions representing a tutorial strategy which aims to promote reflection, feedback expressions, expressions representing a summary of a sub-dialogue and expressions representing answers to learners' queries.

Concerning the utterances linked to tutorial strategies, the questions corresponding to the steps towards a solution (implicit reflection) or the presentation of a problematic situation (explicit reflection) were hard-coded since their contents highly depend on the exercise instance at hand. For the other tutorial strategies (articulation of sub-skills in implicit reflection, intellectualization and reasoning phases in explicit reflection), the system includes a set of class libraries intended to support frames based definition of the contents of other tutor's utterances. The approach presented in this paper is intended to reflect the application of Gagne's skills. The frames are conceived as fill-in-the-blank forms intended to represent typical questions pertaining to a given skill. These forms must be instantiated by an author using the data of a specific exercise. For example, one way to consider a certain principle consists of linking specific problem data with a general domain principle (its premises or its conclusion). An example of this question pattern is designed as follows:

- Intellectualization: ["What are"|"What is"] + <Facts> + <Contextual utterance>
- Reasoning: link facts and premises/conclusion of a principle.
  - ["Are these"]+ <Verb representing the premise/conclusion of principle>
  - ["What is the result of "]+ <condition representing a premise> + <Facts>

Constructing the tutoring dialogue contents for the first exercise in Prolog-Tutor revealed many different ways of formulating questions to reason about a principle. In order to maintain the choice of a frame based representation, an extensive analysis of several instances of Logic Programming exercises using tutoring dialogues would have been necessary. This step also revealed that a more comprehensive approach which integrates the principles to generate natural language was necessary to use those frames: for example Mann and Thompson's Rhetorical Structure theory (Mann & Thompson, 1988). Therefore, a natural language analysis of the learners' answers would have become necessary as well. The contents of the tutor utterances have been hard-coded as a short term compromise to delay solving the expensive natural language challenges. However, given its long term benefit for the system scalability, the use of natural language in Prolog-Tutor is an important trend for further research.

A six-slot frame is used to represent a question in a Prolog-Tutor tutoring dialogue. It also integrates the three other types of utterances that can be expressed by the tutor. The frames include: (1) the text representing the contents of the dialogue question, (2) the skill which could be verified with that question, (3) in case of a sub-dialogue query, the component of reflective thinking triggered by this question (ER-Prolog-Tutor) or the position of this question in the sub-dialogue for implicit reflection (Prolog-Tutor), (4) a set identifying the expected answers to that question, (5) a set of texts representing the two types of feedback for that question, (6) a set of texts consisting of different types of explanations pertaining to that question (predefined answers to "*Why*", "*How do I answer*", "*I don't understand…*" questions and statements from the learner). The expected answers are encoded on the basis of the type of the corresponding question: specific questions which are formulated in such a way that learners only have to type the exact answer, "`unification` questions" where the set of expected answers represents all the valid operations of that `unification`, "`back-tracking`

questions" where the set of expected answers represents the parameters of the specific `back-tracking` in question.

2. Validation of Hard-coded Questions of Dynamic Resources Tutoring Dialogues

The dialogue questions were created by two lecturers in Logic Programming (one of whom is a co-author of this paper) through an iterative process. For each iteration, the dialogue questions were submitted on a paper-and-pencil form separately to two undergraduate students in Computer Sciences. These students were instructed to annotate each question with appropriate comments if they believed they were not sufficiently clear. Enhancing the clarity of these questions was the main goal of this validation process. The questions were continuously improved, until no more comments were encountered.

Combining Learning Objectives and Learning Resources

CREAM learning sessions are planned dynamically with a Petri network while all versions of Prolog-Tutor involve static planning. Relations between learning objectives, instructional activities and resources are specified manually. Static planning conducts a syntactic analysis (parsing) of the XML representation of a learning session structure. This makes it possible to identify each type of element in a learning session as well as their specific representation in the system interface. One future improvement to this aspect will consist of adding the possibility of exploiting the learners' model and other information in order to support dynamic curriculum planning.

The Learner Module

The learner module integrates two static components (a short-term representation of a learner's knowledge and a learning trace) and two dynamic components (two Bayesian inferences algorithms which make it possible to construct the learner's model). The static component content is stored into text files which are parsed when the learner's interface model is launched. The Bayesian inference algorithms were adapted from the JavaBayes library (Cozman, 2001). Two other algorithms were added to complete the inferences. The first additional algorithm is a response parser which processes each learner's action to ensure it can be read by the Bayesian algorithms. The second additional algorithm completes the operations associated with the "Belief updating" algorithm. Its purpose is to take into account the previous probability of mastery associated to each skill when computing the new probability.

Analyzing Learners' Responses

The parser of the learner's answer must detect whether a learner's response is correct. The Bayesian inference algorithm uses this response characterization as input. The response parser is specific to each exercise as the evaluation of a learner's response is intrinsically linked to the data of a specific exercise. This allows preserving the generality and the reusability of the diagnostic inferences by separating them from the data that is specific to a learning activity. The response parser incorporates three syntactic processing algorithms each intended to analyze answers to a specific type of question as defined in the domain module. The first algorithm processes the answers to specific questions, simply comparing the text representing the learner's answer to each of the expected answers, until a

match is found (pattern matching). The second algorithm processes answers pertaining to `Unification of two predicates.` Indeed, learners use a specific interface to perform `unification` operations in order to: (1) indicate if it is a `(Prolog) success` or `failure,` (2) the corresponding substitutions in case of a `(Prolog) success,` (3) the conflicts responsible for the `failure` in case of a `(Prolog) failure`. Using these inputs, the system processes learners' responses using the answer encoded in the domain module which corresponds to the relevant question. The third algorithm processes answers related to the proofs of the `sub-goals` of a `goal based on back-tracking.` These answers are also entered using a specific interface where learners indicate: the `predicate` (a `fact` representing a `goal` or a `sub-goal`) to which the `back tracking` is applied, the `fact` or the `Horn-clause` of the `knowledge base` from which the search will restart, the sought `variable` if there are more than one in the `predicate`. The system takes this data and processes it in order to compare it with the expected answers encoded in the domain module. Notice that the identification of partially correct answers is not yet supported.

Updating the Learner Model

In order to complete diagnostic inferences related to the learner's skills, the result of the analyzer is transmitted and used as evidence in the Bayesian network associated to the domain model. The entry point of that evidence corresponds to the skill associated with the question whose answer was analyzed. Belief updating is performed by default whereas Belief revision is triggered only if the analysis reveals that the learner's answer is incorrect. The original version of Belief updating computes posterior probabilities of Bayesian Network nodes, without taking into account evidences entered into that network earlier. Indeed, before updating a learner's probability of mastering a skill, previous evidence indicating whether the learner has acquired the skill must be considered. A heuristic is applied to posterior probabilities of a learner's mastery of a skill, every time they are computed, in order to account for the past probabilities. At the Nth updating of the learner knowledge state, the following formula is applied to compute the new probability of mastering each skill:

$$Probability_{LTAc}(\zeta)^{N} = [Probability_{LTAc}(\zeta)^{N-1} * (N-1) + Probability_{CTAc}(\zeta)^{N}]/N$$ where:

- $Probability_{LTAc}(\zeta)^{N}$ is the long term probability of mastering the skill ζ at the N^{th} updating of the learner model.
- $Probability_{CTAc}(\zeta)^{N}$ is the short term probability of mastering the skill ζ at the N^{th} updating of the learner model.

Such formulas are necessary as the Bayesian network used is not temporally dynamic (Reye, 2004): new Bayesian updates do not take into account previous updates and evidence. These formulas are used to consider earlier inferences about each skill, through past interactions with the learner: upon the N^{th} update, the load of past inferences is considered by multiplying the resulting probability ($Probability_{LTAc}(\zeta)^{N-1}$) by (N-1). This value is added to the actual inference or posterior probability ($Probability_{CTAc}(\zeta)^{N}$) before being weighted by the factor representing all the established inferences (N).

The Communication Module

When learners complete exercises, they communicate with the system through structured graphic interfaces which makes it possible for them to provide answers to specific questions. Thus, the system must be able to ask questions and understand learners' answers. Two types of interfaces support the realization of training exercises: a main interface and a set of secondary interfaces which are launched only if a specific need comes up. The main interface supports the development of a tutorial dialogue. The secondary interfaces are customized in order to execute specific operations. For example, Figure 7 depicts a customized interface which allows a learner to articulate a `unification` operation as a step towards the `proof of a goal using resolution` in a Logic Programming exercise. Figure 5 illustrates the main interface of Prolog-Tutor. The main exercise statement (question) is printed on the top left-hand side of the main interface. The data that might be used to construct a solution are printed on the top right-hand side of the main interface. In Figure 4, a `Prolog knowledge base` is used by learners to construct the solution to a problem related to the `proof of a goal using resolution` in Logic Programming. The bottom part of that interface has three components. The contents of the tutoring dialogue (tutor's questions, learner's answers) are printed on the center part. The left and right bottom sections of the interface provide learners with the tools they need to communicate with the system. This section describes the communicative acts that evolve in Prolog-Tutor using its main interface.

How Does Prolog-Tutor Initiate Dialogues with Learners?

In order to *maintain the focus* towards the pedagogical goal of acquiring a skill, the tutor controls the interaction with learners while they solve exercises. The tutor triggers the dialogue by printing the first tutorial dialogue question taken from the pedagogical module as explained in the following section. The tutoring dialogue questions are printed on the bottom center section of the exercise interface (Figure 4). Each question put forth to the learner is followed by a legend indicating the elements of the main interface that can be used to provide an answer.

How Can Learners Answer Questions?

Learners answer questions through the commands on the bottom right-hand side of the Prolog-Tutor main interface. Using the "*Direct Answer*" command, they formulate specific answers to specific questions (defined in the domain module). For example, the question "*What is the next Horn-clause relevant for this proof?*" is specific and the learner is instructed to type in the answer directly, using the "Direct Answer" command. Using the "*Result*" command, they articulate the substitutions associated with a `unification which succeeds by default`. Indeed, many questions in Prolog-Tutor – especially in sub-dialogues – require learners to only compute specific `variables substitutions`. The "*Result*" command loads an interface presenting all of the `variables` that should be substituted in a `unification` operation and which queries learners to enter the appropriate `substitution values`. Answering some questions often necessitates other specific interfaces. In that case, a blue button is appended by the system after printing such questions (see Figure 7, last question of the tutoring dialogue). Learners use that button in order to answer using multiple choices forms, specific interfaces supporting the articulation of `backtracking` operations,

etc. Finally, learners can select the "*I do not know*" command to indicate that they are unable to answer a question.

How Does Prolog-Tutor Provide Feedback?

When a learner's answer is correct, the system generates positive feedback and continues the tutoring dialogue. When an answer is not matched with the predefined set of expected answers, the system yields negative feedback and provides learners with a possibility to access two explanations: a general explanation followed by a contextual explanation related to the diagnosed skill. Learners are given the choice to consult these explanations by using the "*Receive Explanation*" command (see Figure 4). The general explanation addresses the default properties of the skill in question. The contextual explanation describes the use of that skill in the particular context of the sub-problem associated to the corresponding DQ question. When learners fail to answer correctly, the explanations are not provided immediately in order to leave them the opportunity to decipher meaning by themselves in the subsequent sub-dialogues.

How Does Prolog-Tutor Support Mixed-Initiative Dialogues?

Prolog-Tutor must understand a variety of learners' needs besides those identified by the cognitive diagnosis. Learners can initiate three types of questions to indicate the following facts (Figure 4): that they do not understand a question (using the "*I do not understand…*" command), that they do not know how to use the interface in order to answer a question (using the "*How do I answer?*" command) or that they do not understand the rationale behind the question in the context of the current problem (using the "*Why?*" command). The system uses the domain module code of the corresponding question in order to respond to the learner's request, since such queries from learners contain predefined answers. From a pedagogical perspective, the third type of question is the most important while the other two could be considered as requests for hints. Indeed, another pedagogical goal of using tutoring dialogues is to better understand learners, and to have them acquire or hone a skill. It is therefore important that they always understand the line of reasoning which characterizes the direction of the tutoring dialogue and its relationship with the exercise currently at hand.

Learners can also visualize the `resolution tree` associated to the `proof` that is actually performed by the learner, using the "*Resolution Tree*" command on the bottom left-hand side of the main interface. In that case, the system loads an image which represents that `resolution tree`.

Finally, learners can examine a representation of the learning domain at any time using the "*Domain Model*" command on the bottom right-hand side of Prolog-Tutor main interface (Figure 4). The system loads a hierarchical representation of the Prolog-Tutor domain in terms of skills and sub-skills. The icons representing the skills that support answering the current question are emphasized. Learners can click on an icon associated to a skill to obtain a brief related explanation.

The Pedagogical Module

The pedagogical module uses tutorial dialogues to open learners' reflection towards their own cognitive states. Consequently, it should integrate structures and processes to manage these dialogues. This section describes the dialogue maintenance mechanisms enabled to support reflection in Prolog-

Tutor and in ER-Prolog-Tutor. Tutoring dialogues are managed by a three-layered architecture similar to Zinn, Moore and Core's 3-tier architecture (Zinn, Moore, & Core, 2002):

- The top layer generates a high-level sequential dialogue plan which represents the predefined steps towards building a solution to an exercise.
- The middle layer supports adaptive planning to adjust the top layer dialogue plans to the current tutorial context.
- The bottom layer is responsible for generating tutor's utterances; interpreting learners' responses and establishing their correctness.

These three layers correspond to three levels of interventions: the top layer dialogue plan integrates Logic Programming skills in a sequential solution building process of an exercise; the adaptive algorithms of the middle layer modify the top layer sequential dialogue plan using learners' models, the bottom layer controls dialogue turn-taking in order to manage explicitly reflective interactions in ER-Prolog-Tutor and mixed-initiative interactions in general.

How Does Prolog-Tutor Generate Tutorial Dialogue Plans for a Solution Building Process?

The top layer of the dialogue management system has a predefined sequential plan. This plan represents the flow of the steps that will be initiated in order to build the solution of an exercise. Every predefined high-level plan has only one type of tutorial action which consists of asking questions to the learners. These questions are intended to support training on Logic Programming skills while trying to solve an exercise. Table 4 presents an example of a sequential dialogue plan. For a given exercise, the top layer uses the corresponding representation of dynamic resources in the domain model to construct a sequential dialogue plan.

Table 4
A predefined sequential dialogue plan

```
<predefined-plan id = p1_exercisePL2>
   <ask-question n=q1> <skill=identification of the proof></ask-question>
   <ask-question n=q2> <skill=use Horn Clause></ask-question>
   <ask-question n=q3> <skill = define sub-proofs></ask-question>
</predefined-plan >
```

How Does Prolog-Tutor Use the Learner Model to Adapt Tutoring Dialogue Plans?

The middle layer of the dialogue management system uses information from the short-term learner model, from the long-term learner model and from the bottom layer of the dialogue management architecture to adapt the default sequential plan from the top layer.

The short term model keeps track of the correctness of learners' answers to dialogue questions *in the current training session*. When a learner fails to answer the current question correctly, the adaptive algorithm modifies the predefined dialogue in order to focus reflective thinking on the corresponding skill. This is supported by inserting sub-dialogues in the predefined plan. When implicit reflection is enabled, this sub-dialogue articulates the sub-skills of the skill associated to the incorrect question. When explicit reflection is enabled, the sub-questions prepared for the learner are intended to foster the components of Dewey's reflection. Once again, the representation of dynamic resources in the

domain model is used to extract the appropriate tutorial utterances for this sub-dialogue. Table 5 presents an example of an adapted plan for implicit reflective thinking on the skill "*use a* `Horn-clause`" for the proof of a goal. In Prolog-Tutor specifically (with implicit reflection), the tutor can further diagnose which questions should be addressed in a sub-dialogue. A negative evidence is sent to the Bayesian Network associated with the domain model; the "most-probable-explanation" algorithm is invoked on the JavaBayes library, the sub-skills which best explain the learner's incorrect answer are known. Therefore, the system will ask the learner only those sub-dialogue questions which are associated to a skill identified as one of the most probable explanations of the learner's failure.

The short term learner's model is also used to alter the sequence of the predefined plan. In a given session, once a learner answers correctly a question associated with a given skill, all the subsequent questions which require that same skill will be skipped.

Table 5
An adapted dialogue plan using the short term learner's model

<adapted-plan id = p1_exercisePL2>
<ask-question n=q1> <skill=identification of the proof></ask-question>
<ask-question n=q2> <skill=use Horn Clause></ask-question>
<enable-sub-dialogue =p1_exercisePL2.**q2.sd> </enable-sub-dialogue>**
<ask-question n=q3> <skill = define sub-proofs></ask-question>
</adapted-plan >

The long-term learners' model represents the probability of mastery for each skill considered in the domain model. This part of learners' models is used to modify the sequential flow of questions *from the top layer of the dialogue plan*. This adaptation of the sequential flow consists of skipping questions whose associated skill is estimated as "probably acquired". A skill is "probably acquired" by a learner if the associated probability of mastery in his long term model is superior to a predefined threshold δ. Table 6 illustrates adaptive planning in Prolog-Tutor, based on the long term learner model. The left-hand side column represents the result of a sequential predefined dialogue plan. The second column represents the result of an adapted plan. Question T3 is associated with the skill "*understand* intuition-of-using-Prolog-Rule". This long-term learner model indicates that the probability that the learner has mastered that skill is superior to the predefined threshold. Thus, an adaptive plan drops Question T3 from the predefined sequential dialogue plan. Moreover, the learner succeeded in answering Question T1, which is associated to the skill "*use* a Prolog-Rule" and this information is recorded into the short-term learner model. Question T5 is also associated to that skill. Since the short term learner's model indicates that this skill is likely to be acquired for the time being, an adaptive plan also removes Question T5 from the predefined sequential dialogue plan. Thus, the adapted dialogue plan starts with Question T1 and continues directly with Question T7. However, there remains a problem since Question T7 is not contextualized anymore (its predecessor questions in the predefined sequence were dropped). In order to keep it clear for learners, the system should re-contextualize it as much as necessary, as illustrated in the right-hand side column of Table 6.

Table 6
Adapting a tutoring dialogue using the long term learner's model

Predefined Dialogue Plan	Adapted Dialogue plan without Re-contextualization	Adapted Dialogue Plan with Re-contextualization
[T1] What is the first element of your knowledge base that could be used to prove `grand_father(X,john)` **[S2]** grand_father(X,Y):- father(X,Z), father(Z,Y) **[T3]** Right. If you use this prolog-rule, what would you do next? **[S4]** Solve the new goals father(X,Z) and father(Z,john) **[T5]** Ok. What element of the knowledge base would you use to solve father(X,Z) **[S6]** father(valery,john) **[T7]** Solve this sub-goal! **[S7]** X= Valery, Z=John. **[T8]** Right. And what about father(Z,John) **[S8]** Z= Stephen **[T8]** Not really. The variable Z used in father(X,Z) is the same as the one used in father (Z,John) ….	**[T1]** What is the first element of your knowledge base which corresponds to the GOAL to prove **[S2]** grand_father(X,Y):- father(X,Z), father(Z,Y) **[T7]** Solve this sub-goal! **[S4]** I do not understand	**[T1]** What is the first element of your knowledge base which corresponds to the GOAL to prove **[S2]** grand_father(X,Y):- father(X,Z), father(Z,Y) *First Contextualization* **[T7]** One of your sub-goal has been identified. Solve this goal! **[S2]** I do not understand *Second Contextualization* **[T7]** You have two sub-goals in order to complete this proof. The first is father(X,Z). Solve this goal! **[S2]** I do not understand

Therefore, dialogue plan adaptation requires that a certain number of parameters be predetermined, which can become a laborious task. Let us consider that the tutorial dialogue just asked Question *I* and that according to the learner model information, the following question would be Question *J* (with *J* being equal to *I+2)*. To ensure intelligible questions, the dialogue manager must be able to link Questions *I* and *J* (What happened between both questions?). This bridge requires predefined expressions to allow summarizing the events that occurred between the introductions of Question *I* and *J*. These contextual bridges make it possible to justify Question *J* since it did not directly follow Question *I*. How many bridges are necessary for a dialogue composed of *N* questions? For Question *K* in the tutorial dialogue, *(N-K)* links must be anticipated. Thus for all the questions, a number of contextual links equivalent to $\sum_{i=1,\ldots,(N-1)}$ (N-i) should be anticipated. Moreover, each bridge may consist of several expressions which incrementally reveal the context of the current question. In the example shown in Table 5, the bridge between T3 and T7 contained two (2) contextual expressions. Therefore, if an average of m expressions compose one bridge, this further increases the amount of contextual links required. For this reason, adaptation algorithms were implemented, but their exploitation remains potential given the large quantity of information necessary and the fact that every question is hard-coded in Prolog-Tutor. This issue has been described in a previous work (Tchetagni & Nkambou, 2006).

How Does Prolog-Tutor Manage Turn-Taking in Reflective and Mixed Initiative Interactions?

Prolog-Tutor uses a mechanism analogue to a finite state automaton to manage turn taking. The process is also similar to dialogue games since the intentionality of the tutor interventions have specific meanings. This approach relates to that of STyLE-OLM as each move indicates what it is intended to do and who should do it (Dimitrova, 2003). Our perspective differs however since some moves are intended to explicitly promote reflection; in STyLE-OLM, dialogue moves are intended to support diagnosis interactions and reflection is considered a side effect of such interactions.

The intentional actions that are supported in the dialogue are represented by the following states: *present-problem-with-statement, trigger-fact-identification, link-fact-to-conditions, link-fact-to-conclusion* which support interactions pertaining to explicit reflection; *challenge-on-skill, challenge-on-sub-skill* which support interactions pertaining to higher-level dialogue or to implicit reflection; *ask-how, ask-why, ask-not-understood, help-learner* which support mixed-initiative interactions; *answer-question, feedback-to-answer* which support all types of interaction.

The turn-taking dynamic is based on a set of rules defining the moves that are allowed between the states. These moves are applied to execute the tutoring action represented in the middle layer plan. Every time an action from the middle layer plan is executed, it is then removed from it. For example, if the first action of the middle-layer plan consists of asking a question, the appropriate question is fetched from the domain module and passed to the communication module which posts it on the learner's interface. Then the following moves are allowed only from the learner: *answer-question*, *ask-how, ask-why, ask-not-understood.*

Conclusion

The previous sections describe basic Prolog-Tutor and ER-Prolog-Tutor concepts as well as their actual implementation. Certain of these elements can be analyzed to study their impact on the intended pedagogy of the system. The main goal of these tutorial dialogues is to explicitly stimulate reflection using principles from a theory of reflective thinking. The originality of this contribution is that the basis of the theory also provides principles likely to serve as landmarks to assess this approach to promote reflection. ER-Prolog-Tutor was qualitatively evaluated with a group of graduate and undergraduate students in Computer Sciences at the University of Quebec in Montreal. The remainder of the paper focuses on this study.

QUALITATIVE STUDY OF ER-PROLOG-TUTOR

This paper shows how metacognition through OLM is enabled in a tutoring system, based on the fundamental principles of Dewey's theory of reflective thinking. One of the main advantages of such a principled approach is that a set of formal criteria are naturally available for analysis during an evaluation. A qualitative study of ER-Prolog-Tutor was performed. Its goal was to examine whether these dialogues actually caused reflection according to Dewey's perspective. The extent to which the learners' actions and reactions echo reflective thinking as defined by Dewey was analyzed. Two main aspects were analyzed: the way in which reflection manifests itself in ER-Prolog-Tutor and whether this manifestation corresponds to the features of Dewey's theory of reflective thinking. This section

describes a qualitative study addressing explicit reflection in ER-Prolog-Tutor as well as the results generated.

Methodology: the Think-Aloud Protocol

The events pertaining to reflective thinking are of a cognitive nature. Therefore, studying their manifestations requires a glance at the internal processes which evolve during reflection. The interest of this type of study lies in the description, the characterization and the justification of the nature of some artifacts that are produced in research (Murray, 1993). In that case, qualitative approaches to evaluation are more appropriate, as they allow investigating a phenomenon in order to explain its manifestations (Miles & Huberman, 2003). Considering Murray's (1993) classification of qualitative approaches to ITS evaluation, the current study was designed to observe and qualify a phenomenon. In this particular case, a group of participants was observed and tape-recorded while they performed an exercise in ER-Prolog-Tutor. "Think-aloud" protocols are put into practice as a means to externalize the mental processes which take place while a task unfolds (VanLehn, 1988). In this study, the data collection was preceded by a three-step process: participant selection, application of the participation procedure and presentation of the instrument of evaluation.

Participants

A valid characterization of explicit reflection in ER-Prolog-Tutor necessitates that potential participants know the Prolog language sufficiently well to carry out an exercise with this program. Six students in Computer Sciences, Educational Sciences and Computational Linguistics took part in the experimentation with ER-Prolog-Tutor. Participants were recruited by means of an advertisement indicating the goals and conditions of the experimentation in the Computer Sciences Department of University of Quebec in Montreal. Prior to the experiment, participants' proficiency with respect to Prolog was assessed by the experimenter (herself a lecturer in Prolog Programming). Two participants were undergraduate students in Computer Sciences who had taken an introductory course in Prolog. They demonstrated abilities to understand and apply Prolog concepts and principles, but they were not able to use these abilities to model reasoning problems as well as a Prolog expert would. Their mastery level of Prolog was considered "Excellent"[2]. Another participant, an undergraduate student in Computational Linguistics uses Prolog regularly at work as well as in other programs of his own. His mastery level was assessed as "Expert"[3]. One participant was a Computer Sciences professional who had taken an introductory course in Prolog and uses Prolog at work. His mastery level was assessed as "Expert". One participant was a graduate student in Computer and Cognitive Sciences who had taken an introductory course in Prolog and experienced Prolog only once in the context of that course. His mastery level was established as "Good"[4]. The last participant was a graduate student in Educational Sciences who used Prolog in his research in Linguistics. His mastery level was established as "Excellent".

---

[2] Excellent mastery level: Someone who understands the concepts and principles of Prolog and who automatically knows when and how to use them and how apply them.

[3] Expert mastery level: Someone who can use Prolog as a programming tool for an application in an arbitrary domain where artificial intelligence is necessary: linguistic, medicine, education, etc.

[4] Good mastery level: Someone who knows the concepts and principles of Prolog and who may know how to apply them.

Procedure

The participants were asked to use ER-Prolog-Tutor to realize an exercise related to the proof of `a goal` using `resolution`. The experimentation proceeded in two phases. First a pre-test was administered to all participants. Second, the pre-test was followed by an interaction with ER-Prolog-Tutor. The purpose of the pre-test was to assess the participants' mastery level of Prolog language in the specific case of the proof `of a goal` based on `resolution`. The purpose of the actual interaction with Prolog-Tutor was to externalize the participants' numerous thoughts as they completed the exercise. Therefore, each participant was requested to think aloud while interacting with ER-Prolog-Tutor, while their voices were tape-recorded for further analyses. Externalizing one's own thoughts can be difficult, especially in an individual training context. To provide support to participants during the think-aloud process, they were given a guide suggesting to (try to) read out loud each question in the tutoring dialogue, self-explain the goal of each question, express any feeling of ambiguity with respect to each question, express any fact which – they thought – was related to a previous question, express any disambiguation, express any reflection about Prolog concepts and principles, express – as often as possible – awareness of the pedagogical approach associated with the tutoring dialogue in the system. The recorded sessions of all six participants lasted eight hours.

Instruments

As mentioned earlier, two instances of exercises are actually implemented in ER-Prolog-Tutor. For this study, the most complex exercise was used. This exercise pertains to the `proof of a goal using resolution` which results in a (Prolog) `Success`, but only after operating a `back-tracking` in the Prolog `knowledge-base`. The main instrument for this evaluation is the tutoring dialogue of ER-Prolog-Tutor associated to that exercise.

This dialogue comprises a set of questions asked to the learner. Each question in such a dialogue has two levels of interactions, labeled D1 and D2. Level D1 supports the incremental construction of a solution to the exercise. It also allows the system to directly diagnose learners' difficulties since the corresponding question is explicitly associated to a skill. Level D2 is a sub-level of D1 that is triggered if a learner fails to answer Level D1 correctly. Questions in Level D2 echo the components of reflective thinking according to Dewey. Their purpose is to stimulate an explicit reflection on the skill that was diagnosed at the corresponding Level D1. In this particular experiment, four diagnosis questions were designed at Level D1 of the tutoring dialogue. Each of these four questions contained a sub-dialogue. All four sub-dialogues comprise a different number of questions to explicitly foster reflective thinking at Level D2. Table 7 presents the second question at Level D1 (Table 7, Q1) of the tutoring dialogue used in this experiment and the corresponding sub-questions at Level D2. In the predefined sub-dialogue corresponding to Q1, three components of reflective thinking are targeted at Level D2: presentation of a problematic situation (Table 7, Q1.S), intellectualization (Table 7, Q1.Intellect1, Q1.Intellect2), asserting a conclusion based on reasoning (Table 7, Q1.Assert1). One notices that in this interpretation of Dewey's theory, four questions were necessary to externalize three components of reflective thinking. Thus, the number of questions at Level D2 will rely on the interpretation of Dewey's theory for the specific skill associated to the question at the corresponding Level D1.

PRESENTATION AND ANALYSIS OF THE RESULTS

According to Miles and Huberman (2003), qualitative data analysis unfolds into three main steps after data collection: coding, compilation and analysis. This section explains how the gathered data was coded, compiled and analyzed in order to relate the participants' mental behavior to reflective thinking as defined by Dewey.

Coding Data

In order to detect some patterns of reflective thinking while the participants answered the D2 level dialogue questions, the first task consisted of relating each of these questions to a step of reflective thinking and labeling it accordingly. The second task consisted of analyzing the tapes recorded during the experiment. A set of codes was established a priori and ad hoc by the authors in an attempt to characterize some manifestations of Dewey's main stages of reflection.

Table 8 presents an excerpt of the participants' answers to the questions asked by the tutor during the sub-dialogue presented in Table 7 (the participant is labeled as P_1 in Table 8).

The last column indicates the codes that best represented the nature of the participant's answer, according to the researchers. In that same column, the codes which appear in italics point to a behavior that differs from the one that the corresponding dialogue question aimed to promote. According to this coding, the first participant demonstrated the following behaviors: he directly responded to the presentation of a problematic situation without any reaction of surprise or confusion (RDC code, Line Q1.S); he clearly identified the key facts of the problem underlying that situation, even if he had questioned these facts previously (OBS_Q code, Line Q1.Intellect; OBS_E code, Line Q1.Intellect2); he provided a final answer to the situation by discussing the facts rather than formally referring to some domain principle, law, concept or acknowledged heuristic of reasoning (R_Arg code, Line Q1.Assert). Moreover, other reactions were noticed even if the dialogue questions were not intended to externalize them. At Line Q1.Intellect (intended to bring out the relevant facts necessary to prove the `goal has_illness(peter,M)`, a participant stated that "*one should verify that the facts symptom(X,S) and cause(S,M) are both true*". This declaration relates to the principle specifying that "*the proof of a goal using a Horn clause consists in the proof of the facts included in its tail*" to the data of the exercise currently in hand. The R_E code is used to label behaviors indicating that a participant is binding a Prolog principle with the data of the current exercise. In conclusion, the first participant's recorded information for this sub-dialogue of the experiment was encoded as [RDC] ➠ [OBS_Q, R_E] ➠ [OBS_E] ➠ [R_App].

Analysis

Analyzing participants' recorded think-aloud conversations has four main goals. Firstly, we would like to establish the extent to which the tutoring dialogue actually triggered the manifestation of reflective thinking from Dewey's point of view. Therefore, the recordings of all participants are examined in order to identify, for each sub-dialogue question, an event which can be interpreted as an occurrence of the corresponding component of reflection. Secondly, we would like to establish the extent to which the participants are aware of the tutoring approach (that is to stimulate reflection) within the tutoring dialogue and how they benefit from this awareness. Therefore, for each sub-dialogue intended to explicitly stimulate reflection, the recordings are examined in order to identify a reaction indicating

that the participant understands why the system is asking the corresponding questions. Thirdly, we would like to examine behaviors that emerged from the interaction with ER-Prolog-Tutor while not predicted by Dewey's theory. Therefore, for each sub-dialogue question, the recordings are examined in order to identify reactions that do not correspond to the component/step of reflection that it is supposed to stimulate. Lastly, we would like to relate the impact of the implementation of the tutoring dialogue in ER-Prolog-Tutor on the stimulation of reflection as perceived by the participants. Therefore, an informal interview was carried out with each participant. During that period, participants were given the opportunity to express their personal perception of the tutoring dialogue in ER-Prolog-Tutor.

Table 7
Evaluation of ER-Prolog-Tutor[5] with ER: Sample of a Tutoring Dialogue

Sub-Dialogue Number	Components of Reflective Thinking	D1 Level Questions	Code
		Let's continue: Could you indicate the first element of the given knowledge base which can be used for this proof?	Q1
D2 Level Questions for Explicit Reflection			
SD1	Situation	The Prolog `rule "has_illness(X, M): - symptom(X, S), cause(S, M)"` must be used to solve the goal corresponding to the Prolog request ?- `has illness(peter, M).` Do you agree?	Q1.S
	Intellectualization	The Horn clause "`has_illness(X, M):- symptom(X, S), cause(S, M)`" expresses a rule. Why? Identify its conditions and consequences	Q1.Intellect1
	Intellectualization	Let's observe the "consequence" in the Horn clause "`has_illness(X, M):- symptom(X, S), cause(S, M)`". Is it `unifiable` with the `goal` to `resolve`?	Q1.Intellect2
	Reasoning	Could you demonstrate how the two `compound terms` have been `unified`?	Q1.Assert1

Manifestation of Reflective Thinking in ER-Prolog-Tutor

Results from the analysis of the participants' records suggest that the sub-dialogues for explicit reflection in ER-Prolog-Tutor may catalyze reflective thinking. However, the extent to which the manifestation of reflection in ER-Prolog-Tutor corresponds to the intrinsic reflective behavior in Dewey's philosophy remains to be examined in order to characterize the quality of reflection in this

[5] The original questions were in French. The questions presented in Table 4 were originally in French.

tutoring system. Therefore, the meaning of the patterns of the encoded thoughts may provide more insight in to their nature. Table 9 shows the compiled frequencies of occurrence of the defined codes of behavior associated to each component of Dewey's reflection, from the analysis of the tape records. These observed codes are presented in a preferential order in Table 9: since they correspond to various manifestations of each component of reflective thinking, some of them represent these components more meaningfully. *P_i* stands for participant number *i* ,*i* varying from 1 to 6. In the following section, proportions are used as evidence to represent the ratio of occurrences of a specific behavior over all the observed behaviors.

Table 8
Transcribed Excerpt of the First Participant's Record

Question	Transcription of P_1's Answers	Encoding
Q1	[P_1] The first element may be `has_illness(X,M) :- symptom(X,S), cause(S,M)`	
Q1.S	[P_1] Oh ! Yes, I agree since `has_illness(X,M)`… Ah! One can find it in the knowledge base.	RDC
Q1.Intellect1	[P_1] The consequence of this rule? And its conditions? Its consequence? Ok, at least the conditions are `symptom(X,S)` and the `causes (S,M)`. The consequence? Ah! It is true, one should verify that `symptoms(X,S)` and that `cause(S,M)` are both true. **[Reads Prolog-Tutor's feedback]** [P_1] I don't understand this. **[Requests an explanation from the system and reads it]** [P_1] Ah! Ok, now I get the meaning of "consequence".	OBS_Q, *R_E*
Q1.Intellect2	[P_1] Why is he asking me this question? Hummm... because unification is a central operation. I think that the proof of the fact `has_illness(X,M)` will succeed only if symptoms and causes are True. So the rule should be unified with the goal. Ok. Yes, they can be unified.	OBS_E
Q1.Assert1	[P_1] Ok. For the `functors`. The `functors` has_illness and has_illness are identical. The arguments. Ah! We have two arguments in each term so it's Ok. The variables. I will unify X with Peter. M is M and we will keep it as is.	R_Arg

Table 9
Dewey's Reflective Behaviors: Frequencies of Occurrences

Total Number of Times each Reflection Phase Was Stimulated	Code of the Components of Reflection	Number of Occurrences Observed for each Participant						Total Number of Occurrences
		P_1	*P_2*	*P_3*	*P_4*	*P_5*	*P_6*	
Presentation of a problematic/confusing situation								**25**
4	**HT**	0	1	1	0	1	1	**4**
	DAT	0	0	0	0	1	0	**1**
	RDC	4	3	3	4	3	3	**20**
Intellectualization of a problematic situation								**21**
3	**OBS_Q**	2	0	0	1	1	0	**4**
	OBS_E	2	3	3	1	2	3	**14**
	OBS_R	0	0	0	1	1	1	**3**
Reasoning to provide a solution to a problematic situation								**49**
6	**R_Pr**	1	0	0	1	0	1	**3**
	R_E	2	3	2	1	1	2	**11**
	R_App	3	5	5	5	5	5	**28**
	R_V	0	0	0	1	0	0	**1**
	R_Arg	1	1	0	1	0	1	**4**
	R_0	0	0	1	0	1	0	**2**
Patterns of reasoning (Co-occurring codes for reasoning behavior)								**21**
Not Applicable	**[R_Pr, R_App]**	1	0	0	1	0	0	**2**
	[R_E, R_App]	2	3	2	1	1	2	**11**
	[R_Pr,R_E, R_App]	1	0	0	0	0	0	**1**
	Other combinations	2	1	1	1	1	1	**7**

LEGEND	
Code	**Meaning**
HT	• Hesitation and interrogation before answering the question • Interrogation about the goal of a tutoring dialogue question
DAT	• Disagreement with an assertion from the ER-Prolog-Tutor when asking a question
RDC	• Direct response to a tutoring dialogue question (no hesitation, no questioning)
OBS_Q	• Self-questioning about the relevant information related to the question asked: facts of the problem, conditions, and prerequisites for the use or the application of a domain law/principle
OBS_E	• Stating of relevant information on the question asked which is generally the information necessary to build an answer to the question
OBS_R	• Restating a question that has been asked
R_Pr	• Evocating a principle, a law, a heuristic, a concept to answer a question
R_E	• Explanation of the application: a domain principle, a law, a heuristic, a concept to answer a question: relating the intellectualized facts (determined in a phase of intellectualization) to the prerequisite of the principle, of the law, of the heuristic, of the concept
R_App	• Direct use or application of a principle, a law, a heuristic, a concept to answer a question: formulating the answer by only evoking the intellectualized facts
R_V	• Indicating that the premises or implications of a domain principle, a law, a heuristic, a concept are violated
R_Arg	• Formulation of an answer on the basis of a general argumentation without a formal reference to a domain principle, a law, a heuristic, a concept to answer a question
R_0	• Evocating a principle, a law, a heuristic, a concept to answer a question

1. Presentation of a Problematic Situation

As far as the presentation of a problematic situation is concerned, 25 instances of behaviors were encoded in the analysis of the (transcribed) participants' answers to all questions intended to present a problematic and a confusing situation. In Table 9, a *direct answer of the participants* (RDC code) was encoded 20 times from a total of 25 encoded behaviors (4/5 or a ratio of 0.8). Table 9 also shows that 4 of 25 behaviors (a ratio of 0.16) were encoded as manifesting *a participant's disagreement with the way in which the tutor presented a situation* (code DAT). Moreover, 1 of 25 behaviors (a ratio of 0.04) was encoded as a manifestation of *a hesitation from the participant* (code HT) before answering. Thus, there was very little evidence that these situations caused confusion or a controversy, as these behaviors are desirable from the outset of reflection, according to Dewey's theory. This can be explained by two hypotheses. Firstly, the level of complexity of the skills targeted – precisely those skills that are related to the proof of `a goal` using `resolution` – may appear to be easily handled given the relatively advanced mastery levels of the participants. Secondly, the nature of the skills related to the proof of `a goal` using `resolution` is either conceptual or procedural. These skills are generally enabled in well structured problems contexts. Therefore, it would be difficult to define a really problematic/unusual situation in exercises related to this task. In order to solve this question, two main suggestions were proposed by the participants during their debriefing. First, they suggested applying this view of reflection in a domain that is more complex than Logic, richer in contents and in potential real world applications. Likewise, the evaluation of ER-Prolog-Tutor with novices in Logic Programming would also be relevant. Thus, it was also suggested to diversify the ways in which the questions are presented in the tutoring dialogues of ER-Prolog-Tutor at Level D2. For example, a participant noticed that even when targeting basic skills, presenting the solution of an exercise and asking learners to build a reasoning process for that solution may be more likely to create confusion than if questions are asked in a classic way.

2. Intellectualization Phase of Reflective Thinking

Concerning the intellectualization of the situations presented to the participants, a total of 21 behaviors were encoded in the analysis of the (transcribed) participants' answers. Most of the dialogue questions targeting key information in these situations achieved their goal. In Table 9, 14 of 21 behaviors (a ratio of 0.67) were encoded as manifesting an effective *identification of relevant facts of these problematic situations* (code OBS_E). As well, 4 of 21 behaviors (a ratio of 0.19) were encoded as manifesting participants' *interrogations about these facts* (code OBS_Q). Finally, in 3 of 21 behaviors (a ratio of 0.14) *participants repeated the current question or previous questions* in order to get a better grasp of these facts (code OBS_R).While these ratios indicate that the corresponding sub-dialogue questions enabled a real intellectualization, we believe that this intellectualization is useful for learners only if *they are aware* of its purpose with respect to the problematic situation stated earlier. In fact, during the 8 hours of recording, the intellectualization questions appeared too explicit. Participants almost always wondered about the goal of the system in asking those questions. Thus, these questions were so centered on the significant facts of the problem that in certain cases, the participants may not have been able to connect them to the main, upper level purpose of the exercise. The next section on "*Awareness of Reflective Thinking*" addresses the topic of reflection awareness during tutoring dialogues.

3. Reasoning Phase of Reflective Thinking

For the reasoning phase, the participants' reactions to questions aimed at formulating an answer to a problematic situation on the basis of a concept, a principle, a law, or an acknowledged heuristic in the domain were sought. A total of 49 such behaviors were encoded. According to Table 9: 3 of 49 behaviors (a ratio of 0.061) were encoded as the enunciation of a Prolog principle, law or heuristic (code R_Pr); 11 of 49 behaviors (a ratio of 0.22) were encoded as a binding of that principle (concept, law or heuristic) to relevant facts of the current exercise (code R_E); 28 of 49 behaviors (a ratio of 0.57) were encoded as a direct application of that principle, law or heuristic (R_App code); 1 of 49 behaviors (ratio of 0.02) was encoded as an expression of the fact that the premises or the implications related to a domain principle, law, heuristic, are violated (code R_V); 4 of 49 behaviors (ratio of 0.081) were encoded as formulations of an answer on the basis of a general argumentation without a formal reference to a domain principle, a law, a heuristic (code R_Arg); 2 of 49 behaviors (ratio of 0.041) were encoded as direct formulations of an answer without any reference to the intellectualized facts or to a domain principle, a law, a heuristic, a concept (code R_0). Considered individually, these ratios are not meaningful from the perspective of Dewey's Theory. Behavior patterns are sought because they allow the differentiation between the direct application of a principle (or of a law, the use of a concept or a heuristic) and the direct application of a principle after having clearly stated it and related it to the facts that have been intellectualized. From the perspective of Dewey's reflection, this differentiation is significant insofar as the goal is to stimulate founded reasoning. For example, applying a principle after stating it and relating it to the key points of a situation confirms the acquisition of three skills: (1) the knowledge of this principle (code R_Pr), (2) the ability to use it adequately (given the facts associated with the situation, one can recognize that this principle should be used) and (3) the ability to apply it properly (given the facts associated with the situation, one can apply the principle in question in order to generate an answer). However, the direct application of a principle (by directly formulating an answer without relating the facts intellectualized to the principle) does not allow drawing such a conclusion, since it can simply arise from chance or from compiled knowledge.

4. Patterns of Behavior in the Reasoning Phase of Reflective Thinking

The last group of codes in Table 9 (patterns of reasoning) presents the co-occurring codes corresponding to reasoning behaviors. A total of 2 of 49 behaviors (ratio of 0,041) indicate a formal enunciation of a principle (law, concept or heuristic) before applying to the current question (co-occurring codes [R_Pr, R_App]). In fact, only 2 of 28 instances (ratio of 0.071) applications of a principle (law, concept or heuristic) were preceded by a formal enunciation. A total of 11 of 49 behaviors (ratio of 0.22) relate a principle (law, concept or heuristic) pertaining to the situation at hand, without formally enunciating it (co-occurring codes [R_E, R_App]). Indeed, 11/28 (ratio of 0.39) applications of a principle (law, concept or heuristic) were preceded by making a link between it and the current question in the tutoring dialogue. A single behavior of 49 (a ratio of 0.02) was encoded as the application of a principle (law, concept or heuristic) after having related that principle with the corresponding question in the tutoring dialogue and after its formal enunciation (co-occurring codes [R_Pr, R_E, R_App]). Thus, one of 28 instances of application of a principle (law, concept or heuristic) follows its enunciation and its relation with the relevant facts in the current question (ratio of 0.035). This means that 14/28 (a ratio of 0.5) applications of a principle are performed directly without enunciating it or relating it to the corresponding situation. From our perspective, the ultimate goal of

an authentic reflection process is to reason explicitly (by stating a principle for example) on the basis of relevant facts. It was thus essential that the participants formally indicate the basis of their answers (the domain principle, law, concept, or acknowledged heuristic) with respect to the facts that were identified in the phase of intellectualization, hence the importance of reasoning patterns of the form (in order of importance): [R_Pr, *], [R_E, *]. The relative majority of the direct applications or uses of the elements of the domain being learned (for example direct application of a principle without first stating it) can be explained by the participants' proficiency in Prolog. It is possible that their skills had reached a certain degree of compilation (Anderson, 1983), which renders the formulation of an answer automatic, once the key elements of a problem have been identified. Indeed, the think-aloud protocols reveal that if one is able to identify the key points of a situation and formulate a solution by relating these points to some domain element, then learners probably master certain skills even if no concept, law, principle, or heuristic is stated formally.

Awareness of Reflective Thinking

Stimulating learners' reflection is of interest only if they are aware of this, especially when it is enabled through a tutoring dialogue. Indeed, unless the goal of each dialogue question is understood, stimulating reflection may be unfruitful as the learner may spend too much time struggling to understand where the tutor is trying to lead him. In order to understand the extent to which the participants were aware of the pedagogical approach of the tutoring dialogue, the recordings were examined to identify the reactions which can reveal the participants' understanding of the reflective purpose of the sub-dialogues.

First, three main facts were sought as they were considered to attest that a participant understood the goal of a sub-dialogue: (1) appropriate interpretation of the purpose of a dialogue question or of the purpose of the pedagogical approach that they perceive through the tutoring dialogue at Level D2 (R_A_BUT code), (2) appropriate interpretation of the purpose of a dialogue question or of the purpose of the pedagogical approach that he perceives through the tutoring dialogue at Level D2, *after that the experimenter has provided a brief hint in that respect* ($R\_A\_BUT_{Exp}$ code) and (3) inappropriate answer to a question (from the experimenter) regarding the purpose of a dialogue question or the purpose of the pedagogical approach that they perceive through the tutoring dialogue at Level D2 (R_NA_BUT code).

Secondly, we tried to identify the moment where participants started to understand the purpose of the sub-dialogues. This observation is relevant in order to judge the impact of reflection on the training experience with ER-Prolog-Tutor since reflection is initiated by the system rather than by a learner (*a priori)*. We believe that fostering this phenomenon is beneficial only when it is carried out consciously, the learner being conscious of the relevance of each micro-action with respect to the original main goal of the exercise (solving a problematic situation).

An analysis of the transcribed records reveals that most participants are conscious that the goal of the system is to lead them to carry out a reflection process on the skills associated to the problem being solved. Five of six participants appropriately interpreted the goal of each sub-dialogue at Level D2, at least once (4 behaviors per participants from P_1 to P_3 and 5 behaviors per participants for P_4 and P_5 were encoded as R_A_BUT; at least 1 R_A_BUT was encoded per sub-dialogue for each participant). For 4 participants (mastery levels: 2 experts, 1 good, 1 excellent), the experimenter had to intervene at least once (no more than twice) to help them understand the goal of certain questions in the sub-dialogues ($R\_A\_BUT_{Exp}$ code). This distribution may be explained by the relatively high

mastery levels of the participants. They may have been confused by the explicit and detailed questions found in the sub-dialogues. The experimenter's interventions were intentionally vague. They were limited to asking the participant what he thought was the goal of the current question at Level D2. An inappropriate interpretation of the goal of the questions in the sub-dialogues of Level D2 (R_NA_BUT code) was encoded for 3 participants (1 for each mastery level: expert, good, excellent), at least once and at most twice (by the participant whose mastery level was assessed "good"). Since most of the participants ended up understanding the goal of each sub-dialogue, this distribution was explained again by their high proficiency and perhaps by the emphasized explicitness of the corresponding questions.

Learners must be aware of their reflective process as soon as possible in order to maximize its benefits. This fact led us to examine the moments at which the participants expressed or demonstrated their understanding of the pedagogical approach underlying the sub-dialogues of the Level D2. For each sub-dialogue at Level D2, the question name and number for which a participant's answer included a reaction suggesting an understanding of the pedagogical approach to stimulate reflection have been sought (R_A_BUT or $R\_A\_BUT_{Exp}$ codes). The tutoring dialogue of this experiment comprised 4 questions associated with a D2 Level sub-dialogue:

- The first sub-dialogue comprised 3 questions. All the participants understood its reflective goal when answering the third question.
- The second sub-dialogue included 4 questions. The first participant understood its reflective goal at the fourth question while others understood it by the second question (this means that on average, the reflective goal is understood at the second question).
- The third sub-dialogue contained 2 questions and its goal was understood by all participants only when they answered the second question.
- The last sub-dialogue involved 3 questions. Two participants understood its goal by the second (P_4) and first (P_6) questions respectively. The four remaining participants understood the sub-dialogue only when they reached the third question.

These observations suggest that participants became aware of reflective thinking rather late in the course of the sub-dialogues. They mostly completed more than half of the sub-dialogues without understanding the purpose of the sub-dialogue questions. Two characteristics of the sub-dialogues at Level D2 could explain this fact: its structure and content. Concerning *the structure of the tutoring dialogue*, the participants suggested - during the informal interview that followed the experimentation - to introduce each sub-dialogue at the Level D2 with an explicit description of its goal with respect to the global goal of the main exercise. Concerning *the contents of the tutoring dialogue*, the inflated explicitness of the questions for intellectualization may be an important factor. For example, a re-examination of Table 9 indicates that participants really intellectualize the problematic situation which is presented to them in sub-dialogues 14 out of 21 times (code OBS_E: stating relevant information in the situation associated with the question currently asked; a ratio of 0.67 of all encoded behaviors with this respect). The intellectualization phase generally occurs in the middle of the reflective process (Dewey, 1933). Consequently, according to the results above, certain participants intellectualize a problematic situation without necessarily knowing/understanding the purpose of this intellectualization (since most of the time, reflection becomes conscious after the middle of the sub-dialogue). In fact, the corresponding questions are so explicit that they may cause the participants to focus on them rather than to focus on their relationship with the main, upper level purpose of the exercise. One way to remedy this weakness of the content of the tutoring dialogue is to encourage

participants to use the "Why?" command more often in the ER-Prolog-Tutor interface. When this command is used, the system explains the relevance of the current question being asked in the tutoring dialogue as well as its relationship with the original goal of the main exercise.

Unpredicted Characteristics of Reflection in ER-Prolog-Tutor

Various components of reflective thinking surfaced at unexpected moments in the reflective sub-dialogues:

- 3 behaviors were encoded as perplexity (code PX, 2 occurrences) and as hesitation (code HT, 1 occurrence) when faced with a specific question intended for intellectualization.
- 18 behaviors were encoded as self questioning about the facts of a situation (code OBS_Q, 8 occurrences), identification of the facts of a situation (code OBS_E, 6 occurrences), repetitions of the current or of the previous questions (code OBS_R, 4 occurrences). These behaviors were encoded from answers to several questions which were all intended for stating a grounded conclusion.
- 20 behaviors were encoded as the enunciation of a principle (code R_Pr, 1 occurrence), the binding of a principle with relevant facts related to the current question in the sub-dialogue (code R_E, 5 occurrences), the direct application of a principle (code R_App, 14 occurrences). These behaviors were encoded from answers to questions which belong to Level D1.

Such manifestations were not foreseen when designing the tutoring dialogue in ER-Prolog-Tutor. The most frequent of such unexpected manifestations of reflective thinking correspond to intellectualization and reasoning (respectively 18 and 19 occurrences, compared to 2 occurrences only for manifestation of confusion after the presentation of a problematic situation). This can be explained by suggesting that reflective thinking may spontaneously appear in a teaching context centered on an interrogative interaction with the learner, even if the matter studied is not particularly problematic. However, explicit reflection should remain a pedagogical goal, in order to make learners articulate their knowledge, at least to promote a mental analysis of their own cognitive state with respect to a domain learned.

Impact of the Implementation of ER-Prolog-Tutor on Reflective Thinking

All participants of this ER-Prolog-Tutor study were interviewed after interacting with the system. The interviews revealed that certain characteristics of the implementation of ER-Prolog-Tutor influenced their experience and consequently, their perception of the underlying pedagogical approach which underlies the system. Two such characteristics have the most significant impact: the main communication interface of the system and the level of complexity of the exercise selected for the experiment.

Concerning the main interface of ER-Prolog-Tutor, participants suggested making it more representative of a pedagogy that fosters explicit reflection. Concretely, externalizing a hierarchical classification of the questions of the tutoring dialogue, as to emphasize the fact that the questions at Level D2 are sub-questions of questions at Level D1, would be a significant step in that direction. Likewise, enabling more explicit feedback through an emotional agent for example, would enhance communication with the learner. The importance of the system feedback stems from the fact that they

could have been used to explain the goal of sub-dialogues (fostering explicit reflection on a skill diagnosed at the Level D1) without burdening the contents of the interactions.

As for the level of complexity of the exercises used in this experiment, participants indicated that it would be appropriate for a beginner-level learner in Logical Programming. Participants selected for this investigation were highly proficient in this field. Thus, they recognized that the main benefit from their experience is that the pedagogical approach in ER-Prolog-Tutor forced them to articulate their knowledge, revealing significant elements of the domain that they took for granted or that they thought were mastered (implicitly), whereas it was not the case.

Summary of the Analysis and Lessons Learned

Results from the analysis of the participants' records suggest that the sub-dialogues for explicit reflection in ER-Prolog-Tutor may catalyze reflective thinking. However, a certain number of factors need to be studied further in order to confirm an absolute correspondence between learners' behavior while interacting with ER-Prolog-Tutor tutoring dialogues and reflection in the perspective of Dewey's theory.

Concerning the presentation of a problematic situation, there was little evidence that the situations presented in Prolog-Tutor always induced the targeted cognitive conflict needed to trigger reflection. This could be explained by the fact that the corresponding questions were not sufficiently problematic given the participants' level of proficiency in Prolog. Unconventional presentation of questions to learners (for example, providing an answer to an exercise and asking participants to build the corresponding reasoning) and higher level questions (which demand using higher level skills such as analysis, evaluation, creation skills) could promote the presentation of more challenging situations when introducing an exercise.

The behavior patterns observed for the intellectualization phase reveals that the dialogue questions reached their intended goals: the participants identified the relevant facts of the problem at hand. However, in the context of reflection, this explicit identification is useful only if the learner is aware of its purpose which is, in this case, to obtain an explicit representation of a problematic situation by isolating relevant facts. We learned that participants intellectualize problematic situations without necessarily knowing/understanding the purpose of this intellectualization. Questions for intellectualization appeared so explicit that they may have caused the participants to focus on them rather than on their relationship with the main, upper level purpose of the exercise. One way to remedy this weakness of the content of the tutoring dialogue would be to encourage participants to use more often the "Why?" command in the ER-Prolog-Tutor interface. Another way is to introduce each sub-dialogue at the Level D2 with an explicit description of its goal with respect to the global goal of the main exercise.

During the reasoning phase, many observed behaviors correspond to the direct applications or to the direct uses of the elements of the domain being learned (for example the direct application of a principle without even stating it). This can be explained by the participants' proficiency in Prolog in the sense that their mastery level had reached a certain degree of compilation (Anderson, 1983), which renders the formulation of an answer automatic, once the key elements of a problem have been identified. The lessons learned from the results of the study pertaining to the presentation of a problematic situation and to the reasoning phase provide some hints about the variables that could be considered in a future evaluation of ER-Prolog-Tutor. First of all, such an experimental study should involve several groups of learners, each group corresponding to a single level of proficiency with

respect to the logic programming skills involved in the study. This approach would allow examining some hypotheses that may explain or qualify more precisely the results from the current study: (1) the perception of a problematic situation depends of the complexity of the question which in turn is a function of the learner's proficiency level (for example, exercises whose purpose is to learn how to build a `knowledge base` in Prolog from the description of a real world situation in natural language); (2) the pattern of reasoning is more articulated with a less proficient learner. Indeed, the direct application of a domain principle or law was the behavior most frequently observed in the reasoning phase of this study. An experimental study could settle whether more skilled learners express less explicitly the reasoning that support their answers and vice-versa for less skilled learners.

Finally, the implementation of ER-Prolog-Tutor may itself contribute to better promote explicit reflection. Making the goal of ER-Prolog-Tutor more obvious to learners may help them understand the pedagogical approach of the system, thus favouring an earlier awareness of what is going on. The learner interface could be enhanced by explicitly labelling the dialogue moves which implement tutorial tactics that explicitly foster reflection. Along the same line, as pointed out in the previous section, further work on the implementation of ER-Prolog-Tutor is needed especially for integrating exercises which require the use of more advanced skills.

Concerning the impact of the implementation of ER-Prolog-Tutor, the lesson learned concerns the importance of the ergonomic factor and the contents of ER-Prolog-Tutor. Learners' interface in ER-Prolog-Tutor could be enhanced to support an earlier awareness of the underlying tutoring tactics. Considering the contents of ER-Prolog-Tutor, the results lead us to consider the integration of problems or exercises requiring more advanced skills. The more complex an exercise is, the longer it could take to foster explicit reflection while building a solution. An experimental study could then examine the extent to which explicit reflection remains pedagogically meaningful and stimulating for learners no matter how much time is needed to complete the solution to an exercise.

More generally, the analysis presented concerns a study which evolved in an experimental context. ER-Prolog-Tutor was not used and evaluated in a real world educational setting. Implanting ER-Prolog-Tutor in such a context would introduce the evaluation of the educational impact or of the efficiency of fostering explicit reflection. Indeed, the observation of specific reflective behaviors at unexpected moments in the current study suggest that reflective thinking may spontaneously appear in an interactive and interrogative pedagogical context, even if the topic of study is not particularly challenging. A further study should focus on the comparison of the reflection patterns resulting from implicit and explicit reflection respectively. How could the educational benefits from each approach be characterized? Is there a significant difference? If so, what are the advantages of each approach? A long-term experimental study would allow comparing the performance of at least two groups of learners who would each use either ER-Prolog-Tutor or Prolog-Tutor over a long term period.

Limitations of the Study

The study could be considered a preliminary analysis of ER-Prolog-Tutor that allowed a general evaluation of its reflective features. Certain aspects remain to be enhanced for a more compelling evaluation of that system, most of which pertain to the experimental settings.

Many studies point to participants' inability to articulate the reasons behind their actions using a think-aloud protocol. As explained, the participants in this study received general instructions to help them articulate the relevant information. In order to obtain further data, in a more reliable manner, other means for capturing participant's thoughts could be introduced. For example, self-explanation

tools could be used after answering each question in the tutoring dialogue. In a subsequent study, audio-video recordings of participants interacting with the system could also be introduced as an additional source of data to substantiate tape recordings.

The codes used to analyze the tape recordings were established *ad hoc* by the experimenter. In order to state their validity objectively, encoding procedures in qualitative studies must involve several coders. A kappa coefficient measure would then be used as an indication of the level of agreement among the coders, supporting the meaningfulness of the final codes (Carletta, 1996).

The number of participants in this study was obviously small. Further studies are required for several reasons. First, the limited amount of data in this study prevented us from explaining certain observations based on the participants' mastery levels since most of them were relatively proficient. Secondly, the level of difficulty of the exercises in ER-Prolog-Tutor suggests that explicit reflection would be more helpful for novice logic programmer participants. No systematic difference was noticed between the participant with a good mastery level and the participants classified as experts. This may be explained by the fact that for participants with high proficiency in Prolog, ER-Prolog-Tutor is still useful as a tool to help them articulate their knowledge. Third, the efficiency of diagnostic algorithms could be observed on the basis of learners' reactions during tutorial dialogues and during their interactions with concrete representations of their model. A comparative study could be conducted to measure the effect of the tutorial dialogue contents, by resorting to an expert other than the author who designed such contents.

These limitations do not invalidate the present study though. This evaluation contributed to estimating the extent to which reflection happens in ER-Prolog-Tutor. Its results suggest that reflection may surface in the current implementation of the system; however, a perfect correspondence with Dewey's reflection remains to be established or at least characterized. We rather consider that the above limitations provide matter for further studies. The methodology used to evaluate ER-Prolog-Tutor should be improved, at least by overcoming the shortcomings cited above. Other factors that indirectly influence the methodology should be examined as well, namely, the validity of the contents of ER-Prolog-Tutor tutoring dialogues. In this implementation, this validity was assessed by two lecturers in Logic Programming and by one undergraduate student in Computer Sciences. Building validity based on a larger set of actors is suitable for a better context of evaluation. A significant number of participants should be gathered in order to account for the lessons learned from the analysis above. Overcoming these limitations will allow us to substantiate and complete the conclusions of the current study.

RELATED WORK

Open learner modeling approaches have received great attention not only because of their computational advantage, but also because of their potential educational benefits as initiators of learners' reflection (Self, 1990). Learners' opened models as tools to promote learner reflection-on-action are used in several ways. In presentation methods, a learner can examine an overview of his level of mastery on the skills or of his level of coverage of the topics of the domain learned (Bull & Nghiem, 2002; Cimolino, Kay, & Miller, 2003; Mitrovic & Martin, 2002). The expected reflection consists of encouraging the learners to recognize their strength and their weaknesses. Reflection also consists of viewing the model of a peer learner in order to promote collaborative learning (Hansen & McCalla, 2003). Another method of reflection allows learners to view and edit their models in order to

change their contents (Dimitrova, 2002). In this case, reflection is expected because before changing the contents of their models, honest learners will have to contemplate their understanding of the domain. Certain approaches allow learners to modify the contents of their models on the basis of a negotiation with the system. In this case, learners have to justify the changes that they wish to make to their models. The arguments used for this justification are a form of reflection since learners will be increasingly attentive to the elements of the domain that are represented in their models. Besides, tutoring dialogues as the medium of open learner modeling through reflection-in-action have been extensively used in ITSs: with natural language processing components as in: Auto-Tutor (Person, Graesser, Kreuz, Pomeroy, & Group, 2001), Atlas-Andes (Rosé et al., 2001), Geometry-Explanation-Tutor (Aleven, Popescu, & Koedinger, 2003), and with menu-driven components as in CATO (Ashley, Desai, & Levine, 2002), Ms. Lindquist (Heffernan & Koedinger, 2002).

ER-Prolog-Tutor supports OLM through reflection-on-action using a presentation approach similar to (Bull & Nghiem, 2002). However, our method differs from this by suggesting a probability of mastery of a skill as the subject of reflection, rather than a percentage of wrong and of correct answers for each concept. In that perspective, ER-Prolog-Tutor opens the contents of learners' models to foster reflection at a more abstract level, in a way more similar to that of Corbett and Anderson (1995) and that of Zapata-Rivera and Greer (2003). However, these latter systems do not support the modification of learners' models content by a formal negotiation with the system, while ER-Prolog-Tutor does. In our case, that negotiation is enabled by allowing a learner to modify their probability of mastery by solving an exercise focused on that skill. Another formal approach to negotiating learners' model contents is supported in STyLE-OLM (Dimitrova, 2002) where during a dialogue with the system, the learner has to show their mastery level of a concept. While both approaches find their way to promote reflection-on-action, the rationale of negotiation of learners' model contents in ER-Prolog-Tutor is to strengthen their commitments to their own assertions since they will have to support them by performing specific exercises without the help of the system.

The original contribution of this paper carries on its perspective of OLM through reflection-in-action in ER-Prolog-Tutor. Indeed, two issues pertain to the use of tutoring dialogues to promote metacognition through reflection have been addressed in this paper. First, they should be conducted in a coherent manner so that the learner is always aware and focused towards a learning goal or at least, towards a solution. Second, the reflective activities which emerge from these tutoring dialogues are a side effect of their interactive nature. There is not always concrete evidence of their occurrence (Dimitrova, 2003). Indeed these dialogues should not only allow the learner to construct the right solution to a problem, they should also explicitly elicit the skill which underlies the construction of that solution (Aleven et al., 2003). Furthermore, enabling metacognition efficiently requires the explicit modeling and an explicit practice of its elements (Schoenfeld, 1987). None of the tutoring dialogue based systems mentioned above explicitly consider both of these issues at once. Their tutoring dialogues allow learners to build the desired answer. However a question remains, "What happened exactly? Was it assimilation of knowledge, a lucky guess or an explicit insight into the skills to be learned and used?" Auto-Tutor's tutoring dialogues are based on a formal structure of dialogue moves which represent several pedagogical tactics, but which structure fails to warrant explicit reflection on a specific skill. Andes Physics tutor uses knowledge construction dialogues (KCD) to guide the learner towards a solution (VanLehn et al., 2005). However the nature of what is actually processed and the manner in which it is processed are integrated into very specific questions pertaining to a specific skill, *traced* from a task graph in Andes. Learners' reasoning and learners' reflection are not explicitly articulated through the system as ER-Prolog-Tutor strives to do. The

Geometry-Explanation-Tutor supports an intuitive approach to explicit reflection in the context of reflection-on-action: learners self-explain their solution of a geometry problem by describing in their own words how they have applied a principle. Our approach is more formal as it uses Dewey's theory of reflection to guide the modeling of tutoring strategies. Learners are given more freedom in externalizing their reflection process in the former case. ER-Prolog-Tutor carries learners throughout several phases of reflection. We consider that these two perspectives contribute to fostering reflection at different levels: Geometry Explanation Tutor approach could be seen as more appropriate to independent, self-conducted, mature learners; ER-Prolog-Tutor would be more appropriate for novice and less confident learners. Dimitrova's interactive approach to OLM uses a set of dialogue "moves" to support learners' interactions with their model (Dimitrova, 2002). These dialogue "moves" represent communicative acts to stimulate reflection; however, the meaningfulness of these acts is not formally supported by an appropriate theory or by an underlying logic which is aimed towards an explicit reflection as in ER-Prolog-Tutor. Ms Lindquist tutoring dialogue reflects tutoring strategies for representing different approaches in constructing a solution to algebraic problems. While these approaches are interactive, their link to reflection-in-action is not clearly defined as is the case with ER-Prolog-Tutor's strategies. MIRA's reflection assistant approaches reflection from a more global stand-point by supporting the learning of the main metacognitive behaviors: auto-evaluation, self monitoring, planning (Gama, 2004). These behaviors apply to the global learning process while the approach presented in this paper is directed towards reflection on a specific task for some specific types of skills (Gagne's intellectual skills).

Furthermore, a number of empirical studies have investigated reflection in tutoring systems. The evaluation of the MIRA system was concerned with how learners' metacognition changes as a result of interacting with a reflection assistant (Gama, 2004). This experimental study differs from the qualitative study presented in this paper since its goal is to evaluate the benefits of metacognitive activities on the learning experience. Most importantly, the evaluation of the reflection assistant in MIRA focuses on generic metacognitive skills (problem understanding and knowledge monitoring, selection of metacognitive strategies, and evaluation of the learning experience). The evaluation of ER-Prolog-Tutor concerns the reflection process properly without a link with the corresponding learning experience. An evaluation involving Andes Physics tutor investigated whether informal reflection questions led to better conceptual understanding and increased problem-solving ability in Andes Physics Tutor (Katz, Allbritton, & Connelly, 2003). Our qualitative study investigates whether questions explicitly aimed at triggering reflection actually do support reflective thinking. While evidence of reflection may suggest a better conceptual insight and understanding, this question is not explicitly addressed here. Similarly, Woolf and colleagues evaluated the impact of using a general platform for inquiry learning to assess learning on participants' reasoning skills: critical thinking skills and inquiry skills (Woolf et al., 2002). This platform is analogous to the platform of ER-Prolog-Tutor in that it provides tools to learn by emulating explicitly the phases of a scientific inquiry. However, the study of that platform does not question the inquiry process as it is the case for reflection in the study of ER-Prolog-Tutor.

Finally, ER-Prolog-Tutor learners' models are conceived as an overlay built from Bayesian inferences on the domain model which was designed as a Bayesian Network. Bayesian reasoning is a relevant approach to support inferences under uncertainty. Combining domain overlays and Bayesian networks has already been used in learner modeling approaches in ITS. ER-Prolog-Tutor learner model updates are similar to the approaches of Collins, Greer and Huang (1996) and to that of Martin and VanLehn (1995). However, the structure of our network is not hierarchical as in the former case.

As well, there is only one type of node in an ER-Prolog-Tutor network which represents all domain skills (apply a prolog principle, understand a Prolog concept, etc.) while Martin and VanLehn's network differentiates between several kinds of nodes to represent learners' reasoning and learner's mental states as accurately as possible for a specific problem. The diagnosis of learners' errors is similar to the method used by Conati, Gertner and VanLehn (2002) as a heuristic analogous to belief revision is used in that paper. They also used Bayesian networks to form a prognosis of learners' reasoning and actions while problem solving. However, in that case, prognosis was intended to support coached problem solving and in ER-Prolog-Tutor, this pedagogical philosophy is replaced by the stimulation of reflective thinking.

CONCLUSION

This paper presented two philosophies for making learners aware of their own cognitive state, using the conceptual features of Prolog-Tutor, an ITS for Logic Programming. The first philosophy corresponds to OLM based on reflection-on-action where learners consult the contents of their model in an ITS. The second philosophy is substantiated by OLM based on reflection-in-action. In order to enhance the quality of learner's interactions with their models (which is reflection-on-action based OLM), reflection-in-action based OLM is presented as a preliminary stage which allows them to become aware of the knowledge elements of the domain learned and most importantly, of their cognitive state with respect to these elements. After a general presentation of Prolog-Tutor, the paper focused on reflection-in-action based OLM through the introduction of tutoring dialogues that explicitly promote reflection in that system (ER-Prolog-Tutor). Dewey's components of reflective thinking were used to define strategies as the goals of questions for reflection: using Drake's analogy between critical thinking and reflection, we were able to specify these strategies more precisely with respect to each type of skill found in the Gagne's taxonomy. Based on the goals associated to these tutorial strategies, the tutorial tactics for explicit reflection were defined according to one type of Gagne's skills. These tactics were implemented as the contents of questions that are asked in the explicitly reflective sub-dialogues. As these questions refer to the nature of the skill on which the learner reflects (as defined by Gagne's taxonomy) and to the nature of reflection (as defined by Dewey's theory). An advantage of that approach to OLM is that the underlying strategies and tactics are transferable to other learning domains, as long as these domains' contents are expressed in terms of skills derived from Gagne's skills taxonomy. ER-Prolog-Tutor dialogues are original as their design is based on Dewey's theory of reflective thinking, in order to explicitly promote OLM through reflection-in-action. AIED researchers in artificial intelligence in education have often pointed out that when tutorial strategies are not based on a specific learning or teaching model, they are difficult to justify, to select, to reflect upon and to improve (Ford, 1987; Mizoguchi & Bourdeau, 2000). This contribution proposes an answer to that issue in the context of OLM through reflection-in-action. Another rationale for the principled approach which characterizes our contribution is that a set of formal criteria are naturally available for analysis during an evaluation.

A formal qualitative study of ER-Prolog-Tutor was conducted using think-aloud protocols. Four main points were analyzed: (1) the characterization of the manifestation of Dewey's reflection process when interacting with its tutoring dialogue for explicit reflection, (2) the importance of being aware of thinking reflectively while interacting with its tutoring dialogue, (3) a characteristic of reflection that is not predicted in Dewey's theory of reflection, and (4) the impact of the implementation of ER-

Prolog-Tutor on the benefit of stimulating reflection through it. The participants' think-aloud protocol recordings enabled the identification of patterns of mental behavior witnessing the occurrence of reflective thinking according to Dewey. The study shows that the intellectualization and reasoning components of Dewey's reflection actually took place as in Dewey's philosophy, while the component related to the presentation of a problematic situation did not really cause confusion to the participants. The participants' proficiency in Logic Programming was presented as the best explanation for that observation. The participants ended up understanding the pedagogical purpose of the sub-dialogues of ER-Prolog-Tutor, although it occurred generally late in the course of each such sub-dialogue. Despite the formal structure of the sub-dialogues of ER-Prolog-Tutor, reflective thinking did not always appear according to this model. The interrogative nature of a tutorial interaction may enable a component of reflection, even if it is not planned in that particular interaction. One of the main enlightening aspects of the study is that the relevance of reflective thinking may depend on the level of proficiency by the learner, with respect to the skill to which that reflection relates, as well as the level of complexity of the training context in which it is held. This introduces two main research questions namely, how reflective tutoring dialogues (especially those integrated in ER-Prolog-Tutor) could be improved to foster the awareness of reflection as early as possible, and how reflective thinking could be explicitly stimulated (based on Dewey's theory) for higher level skills (such as the construction of a `knowledge base` in Prolog, given the description of a corresponding situation). These questions are the main research directions that this evaluation has indicated for future work.

REFERENCES

Aleven, V., Popescu, O., & Koedinger, K. (2003). Towards tutorial dialog to support self-explanation: adding natural language understanding to a cognitive tutor. In U. Hoppe, F. Verdejo & J. Kay (Eds.) *Artificial Intelligence in Education, Proceedings of the 11th International Conference, AIED2003* (pp. 39-46). Amsterdam: IOS Press.

Anderson, J. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Ashley, K., Desai, R., & Levine, R. (2002). Teaching case-based argumentation concepts using dialectic arguments vs. didactic explanations. In S. Cerri, G. Gouardères & F. Paraguaçu (Eds.) *Intelligent Tutoring Systems, Proceedings of the 6th International Conference, ITS2002* (pp. 585-595). Berlin: Springer.

Bull, S., & Nghiem, T. (2002). Helping Learners to Understand Themselves with a Learner Model Open to Students, Peers and Instructors. In P. Brna & V. Dimitrova (Eds.) *Intelligent Tutoring Systems. Proceedings of Workshop on Individual and Group Modelling Methods that Help Learners Understand Themselves* (pp. 5-13).

Bull, S., & Pain, H. (1995). Did I say what I think I said, and do you agree with me?: inspecting and questioning the Student Model. In J. Greer (Ed.) *Proceedings of World Conference on Artificial Intelligence in Education* (pp. 501-508). Charlottesville, VA: AACE.

Bull, S., McEvoy, T., & Reid, E. (2003). Learner models to promote reflection in combined desktop PC/Mobile Intelligent learning environments. In U. Hoppe, F. Verdejo & J. Kay (Eds.) *Proceedings of Workshop on Learner Modelling for Reflection*, Supplementary Proceedings of the 11th International Conference, Volume V, AIED2003, (pp. 199-208).

Carletta, J. (1996). Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics,* 22(2), 249-254.

Cimolino, L., Kay, J., & Miller, A. (2003). Incremental Student Modelling and Reflection by Verified Concept Mapping. In S. Bull, P. Brna & V. Dimitrova (Eds.) *Proceedings of Workshop on Learner Modelling for Reflection*, Supplementary Proceedings of the 11th International Conference, Volume V, AIED2003 (pp. 219-227).

Collins, J. A., Greer, J. E., & Huang, S. X. (1996). Adaptive assessment using granularity hierarchies and bayesian nets. In C. Frasson, G. Gauthier & A. Lesgold (Eds.) *Intelligent Tutoring Systems, Proceedings of the 3rd International Conference, ITS96* (pp. 569-577). Berlin: Springer.

Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *User Modeling and User-Adapted Interaction,* 12(4), 371-417.

Corbett, A., & Anderson, J. (1995). Knowledge tracing: modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4, 253-278.

Cozman, F., G. (2001). The JavaBayes system. *The ISBA Bulletin,* 7(4), 16-21.

Cumming, G., & Self, J. A. (1991). Learner modeling in collaborative intelligent educational systems. In P. Goodyear (Ed.) *Teaching Knowledge and Intelligent Tutoring* (pp. 85-104). New Jersey: Ablex Publishing Corporation Norwood.

Dewey, J. (1933). *How We Think: A Restatement of the Relation of Reflective Thinking to the Educative Process*. Boston: Heath & Company.

Dimitrova, V. (2002). STyLE-OLM: interactive open learner modeling. *International Journal of Artificial Intelligence in Education*, 13, 37-58.

Dimitrova, V. (2003). Diagnostic interactions that promote learner reflection. In S. Bull, P. Brna & V. Dimitrova (Eds.) *Proceedings of Workshop on Learner Modelling for Reflection*, Supplementary Proceedings of the 11th International Conference, Volume V, AIED2003 (pp. 228-237).

Drake, A., J. (1976). *Teaching Critical Thinking: Analyzing, Learning and Teaching Critical Skills*. Danville: Interstate.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: a new area of cognitive-developmental inquiry. *American Psychologist,* 34, 906 - 911.

Ford, L. (1987). Teaching strategies and tactics in intelligent computer aided instruction. *Artificial Intelligence Review,* 1(3), 202-215.

Gagne, R. (1992). *The Conditions of Learning and Theory of Instruction*. New York: Holt, Rinehart & Winston.

Gama, C. (2004). Metacognition in interactive learning environments: the reflection assistant model. In F. Paraguaçu, J. C. Lester & R. M. Vicari (Eds.) *Intelligent Tutoring Systems, Proceedings of the 6th International Conference, ITS2004* (pp. 668-677). Berlin: Springer.

Hansen, C., & McCalla, G. (2003). *Active Open Learner Modelling.* In S. Bull, P. Brna & V. Dimitrova (Eds.) *Proceedings of Workshop on Learner Modelling for Reflection*, Supplementary Proceedings of the 11th International Conference, Volume V, AIED2003 (pp. 248-257).

Heffernan, N., & Koedinger, K. (2002). An intelligent tutoring system incorporating a model of an experienced human tutor. In S. Cerri, G. Gouadères & F. Paraguaçu (Eds.) *Intelligent Tutoring Systems, Proceedings of the 6th International Conference, ITS2002* (pp. 596-608). Berlin: Springer.

Katz, S., Allbritton, D., & Connelly, J. (2003). Tutoring that takes place after a problem has been solved. *International Journal of Artificial Intelligence in Education,* 13, 79-116.

Kolb, D. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs, NJ: Prentice-Hall Inc.

Lewin, K. (Ed.). (1948). *Resolving Social Conflicts: Selected Papers on Group Dynamics*. New York: Harper & Row.

Mann, W., & Thompson, S. (1988). Rhetorical structure theory: toward a functional theory of text organisation. *Text,* 8(3), 243-281.

Martin, J., & VanLehn, K. (1995). A bayesian approach to cognitive assessment. In P. Nichols, S. Chipman & R. L. Brennan (Eds.) *Cognitive diagnostic assessment* (pp. 141-165). Hillsdale, NJ: Erlbaum.

Mayo, M., & Mitrovic, A. (2001). Optimising ITS Behaviour with Bayesian Networks and Decision Theory. *International Journal of Artificial Intelligence in Education*, 12, 124-153.

Miles, M. M., & Huberman, A. M. (2003). *Analyse des données qualitatives, traduction de la 2e édition américaine.* Bruxelles: De Boeck.

Mitrovic, A., & Martin, B. (2002). Evaluating the Effects of Open Student Models on Learning. In P. De Bra, P. Brusilovsky & R. Conejo (Eds.) *Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 296-305). Berlin: Springer-Verlag.

Mizoguchi, R., & Bourdeau, J. (2000). Using Ontological Engineering to Overcome Common AI-ED Problems. *International Journal of Artificial Intelligence in Education,* 11, 107-121.

Moser, M., & Moore, J. (1996). Towards a synthesis of two accounts of discourse structure. *Computational Linguistics,* 22(3), 410-419.

Murray, T. (1993). Formative qualitative evaluation for "exploratory" ITS research. *Journal of Artificial Intelligence in Education,* 4(2/3), 179-207.

Nkambou, R., Frasson, C., & Gauthier, G. (2003). CREAM-Tools: An authoring environment for knowledge engineering in intelligent tutoring systems. In T. Murray, S. Blessing & S. Ainsworth (Eds.) *Authoring Tools for Advanced Technology Learning Environments: Toward cost-effective adaptive, interactive, and intelligent educational software* (pp. 269-308). Dordrecht: Kluwer Academic Publisher.

Ohlsson, S. (1987). Some principles of intelligent tutoring. In R. W. Lawler & M. Yazdani (Eds.) *Learning Environments and Tutoring Systems* (Vol. 1, pp. 203-237). Norwood, New Jersey: Ablex Publishing.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA.: Morgan Kaufman Publishers.

Person, N., Graesser, A., Kreuz, R., Pomeroy, V., & Group, T. R. (2001). Simulating human tutor dialog moves in AutoTutor. *International Journal of Artificial Intelligence in Education*, 12, 23-39.

Reye, J. (2004). Student modeling based on belief networks. *International Journal of Artificial Intelligence in Education,* 14, 63-96.

Rosé, P., Jordan, P., Ringenberg, M., Siler, S., VanLehn, K., & Weinstein, A. (2001). Interactive conceptual tutoring in Atlas-Andes. In J. D. Moore, C. L. Redfield & W. L. Johnson (Eds.) *Artificial Intelligence in Education, Proceedings of the 10th International Conference, AIED2001* (pp. 255-266). Amsterdam: IOS Press.

Schoenfeld, A. H. (1987). What's all the fuss about metacognition? In A. H. Schoenfeld (Ed.) *Cognitive Science and Mathematics Education* (pp. 189-215). Hillsdale: Lawrence Erlbaum Associates.

Schön, D., A. (1983). *The Reflective Practitioner. How Professionals Think in Action*. London: TempleSmith.

Self, J. (Ed.). (1990). *Bypassing the intractable problem of student modelling*. Norwood, NJ: Ablex.

Tchetagni, J., & Nkambou, R. (2006). Elaborating the Context of Interactions in a Tutorial Dialog. In R. Dapoigny & A. Moonis (Eds.) *Advances in Applied Artificial Intelligence, 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE06* (pp. 848-858). Berlin: Springer.

Tchetagni, J., & Nkambou, R. (2004). Diagnosing learner errors in a web based environment using the MPE. In V. Uskov (Ed.) *Web Based Education, Proceedings of the 3rd IASTED Conference, WBE2004* (pp. 135-145). Calgary, Canada: ACTA Press.

Tchetagni, J., Nkambou, R., & Bourdeau, J. (2005). Supporting student reflection in an intelligent tutoring system for logic programming. In J. Kay, A. Lum & J.-D. Zapata-Rivera (Eds.) Artificial Intelligence in Education, 12th International Conference, AIED2005, Proceedings of the Workshop on LeMore (pp. 42-51).

VanLehn, K. (1988). Student modeling. In M. Polson, C. & J. Richardson (Eds.) *Foundations of Intelligent Tutoring Systems* (pp. 55-78). Hillsdale: Laurence Erlbaum.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., et al. (2005). The Andes physics tutoring system: lessons learned. *International Journal of Artificial Intelligence in Education,* 15(3), 147-204.

Vassileva, J., McCalla, G., & Greer, J. (2003). Multi-agent multi-user modelling in I-Help. *User Modeling and User Adapted Interaction,* 13(1), 179-210.

Woolf, B., Reid, J., Stillings, N., Bruno, M., Murray, D., Reese, P., et al. (2002). A General Platform for Inquiry Learning. In S. Cerri, A., G. Gouardères & F. Paraguaçu (Eds.) *Intelligent Tutoring Systems, Proceedings of the 6th International Conference, ITS2002* (pp. 681-697). Berlin: Springer.

Zapata-Rivera, J.-D., & Greer, J. (2003). Analyzing student reflection in the learning game. In S. Bull, P. Brna & V. Dimitrova (Eds.) *Proceedings of Workshop on Learner Modelling for Reflection*, Supplementary Proceedings of the 11th International Conference, Volume V, AIED2003 (pp. 288-298). Amsterdam: IOS Press.

Zinn, C., Moore, J., & Core, M. (2002). A 3-Tier planning architecture for managing tutorial dialogue. *Intelligent Tutoring Systems, Proceedings of the 6th International Conference* (pp. 574-584). Berlin: Springer.