

LeVinQam: A Question Answering Mining platform

Patrick Duval, Agathe Merceron, Christian Rinderknecht, Michel Scholl

ESILV - Génie Informatique

Pôle Universitaire Léonard de Vinci

F-92916 Paris La Défense - Cedex

FRANCE

{patrick.duval, agathe.merceron, christian.rinderknecht, michel.scholl}@devinci.fr

The development of Web technologies has accelerated the offer of (commercial) e-learning platforms. This paper is a first step toward the design and implementation of a platform called LeVinQam whose main functionalities are (a) the provision and management of a series of exercises and tests (authoring tools), (b) a personalized navigation through the existing collection of exercises and tests, taking into account the user profile (level of skills) and history, (c) the analysis of user answers, its storing into a database and its mining. As case studies we have chosen (1) the learning of SQL, the standard database query language, and (2) formal proofs for propositional logic, to validate the platform functionalities and the mining of answers.

I. INTRODUCTION

The development of Web technologies has accelerated the offer of (commercial) e-learning platforms. However most of the current systems still provide a limited set of functionalities in terms of generic authoring tools, translation tools for importing existing course material, answer analysis tools, comprehensive collection of answers for future mining, etc.

This paper is a first step toward the design and implementation of a platform whose main functionalities are (a) the provision and management of series of exercises and tests (authoring tools), (b) a personalized navigation through the existing collection of tests and exercises, taking into account the user profile (level of skills) and history, (c) the analysis of user answers, its storing into a database and its mining. We follow what is called a process oriented approach in [7].

The focus of the paper is on the design of a generic platform, emphasizing three aspects: (1) information model, (2) platform architecture and (3) visualisation and mining of users' results. The system we are aiming at should provide rich feedbacks to both learners and teachers and be open enough to ease evolution and developments from both course authors and software developers.

As a case study, we have chosen the learning of SQL, the standard database query language, to validate the platform functionalities related to navigation. Then, to illustrate our approach for students' answers mining and visualisation, we chose a mathematical reasoning case study (formal proofs in propositional logic), for which automatic answer checking and error detection has been implemented in a web-based tool, the Logic-ITA [1]

The contribution of this paper is threefold:

- A rich information model with four components: (i) exercises model which allows for a generic structuration (the internal structure of exercises and tests can be chosen by the author); (ii) a model for *e-learning guided tours* (egt) which are subsets of exercises that define consistent sequences of exercises, similar to sequencing [4], egt can equivalently be seen as a navigation with constraints through the set of exercises existing in the database; (iii) a personalized navigation and choice of exercises by the user (driven by the teacher that allows only specific tours but also taking into account the user former scores and history); (iv) storing into the database the user scores, answers and errors. The database may also include information about teachers and documents describing exercises and predefined answers.
- The design of a web platform called LeVinQam currently under development. The main objective is to provide a core platform with an open design, offering efficient extension mechanisms based on robust and open standards like XML and java. This in order to ease contributive developments from both authors and software developers, and to allow researchers to capitalize and share more effectively on e-learning experiments.
- Tools to visualize and mine students' answers. Having all students answers and errors at hand makes it possible to use queries and data mining techniques to retrieve pedagogically relevant information for both students and teachers [2]. Teachers often ask for techniques that cluster students in homogeneous groups. A way to achieve this is to use a Data Mining technique called clustering. Histograms prove to be useful to convey back to teachers information about the obtained clusters. Also mistakes made by students may be mined to find whether some errors are often made together. We illustrate the approach using students answers from the Logic-ITA [1].

Section 2 defines the e-learning information model we choose with emphasis on the modeling of egt. The design of the LeVinQam platform is addressed in Section 3. Section 4 presents some benefits of the mining and visualisation approach.

II. E-LEARNING INFORMATION MODEL

We define in this section the information model of LevinQam. We illustrate it on the learning of SQL and use the following scenario.

A. SQL e-learning scenario

Assume the student has an unconstrained access to an on line course material on SQL. The course can have any structure. For example it might be a sequence of 4 chapters, namely introduction, relational model, relational algebra, and relational calculus. The decomposition of chapters into sections, subsections is not further detailed.

The student has a constrained access to exercises. Again exercises can have any structure. As an example, an exercise is a sequence of questions. Questions can have a grade associated with.

Exercises can be linked to one or several pieces of the course at any granularity level. For instance, a set of exercises is associated with *selection*, a section of chapter *relational algebra*.

The teacher may define the following two level egt.

- 1) There are four sets of exercises, one for each chapter of the course. Prior to taking an exam (a set of graded exercises), the students must pass all four sets of exercises in any order. Students can spend as much time as they want on any exercise.
- 2) Pass a set of m exercises associated with a chapter implies passing $n < m$ exercises¹. The choice of the n exercises among the m available exercises is left to students.

No further assumption is made on the level of the student and history of the student.

Students' answers are stored in the database with information about students, exercises, time, mistakes made, etc.

The teacher can not only visualise at which step a given student currently stands, global snapshots of the classroom (e.g. how many students currently passed the relational calculus step) but also can visualise some grouping of the students according to the errors they made when answering.

B. Model

An original feature of the model is that given an e-learning material (course or exercise) with a given structure, it allows for (1) an elegant and powerful specification by the teacher of constraints on the access by the student to items of this e-learning material, (2) a guided traversal of this material by the student depending on his/her history, (3) the logging of student actions for further data mining and (4) tools for a synthetic view (snapshots) of the state of progress of a set of students.

Although the model should apply to a large variety of pedagogical situations, such as management of the state of progress of a student through a university cursus, detailed course delivery, training for an eventual exam on a given course material, etc., without loss of generality we focus here on the latter case as illustrated in the previous scenario.

The information model includes three components: (1) exercise and course modelling, (2) guided navigation through this material and (3) student log. We successively define these three components. In the last subsection the functional architecture of the system which relies on these three components is sketched.

¹passing an exercise which is not further detailed might be just writing an answer, or writing an answer which is graded by the teacher, or automatically graded, etc.

e-learning material modelling: We assume that the e-learning material is a document which respects an hierarchical structure in which leaves represent atomic multimedia components that can be displayed on any Web environment. We assume the material is representable by an XML document whose structure can be specified and controlled by standard XML typing mechanisms. We do not make any further assumption on the structure and granularity. As in the previous scenario, a course on SQL may be a sequence of chapters (introduction, relational model, relational algebra, SQL, etc.). The chapter on SQL may be a sequence of sections (unary operators, binary operators, aggregate functions, etc.). The section on unary operators may be further subdivided into an atomic item on the selection operator and another on the projection operator. The lower level detailed structure of an atomic item (inclusion of drawings, figures with text, display onto Web pages, etc.) is out of the scope of this paper. Similarly, a set of exercises is modeled by an XML document.

An exercise (a set of exercises) is associated with a piece of course (leaf or subtree) by a node to node mapping in the tree representation of the course and exercises documents. Given a course document and an exercise document several mappings might be defined. Furthermore a set of exercises might be of interest for several parts of a given course, for different courses.

e-learning guided tour: On-line course materials may be toured in different ways.

- *On-line documentation.* Access is restricted to reading the course items. No constraint exists on the access to any item of the course. No order on the successive items to be accessed is either specified, as for the course material on SQL in the previous scenario.
- *Linear order.* In contrast, consider a set of courses given in a school cursus that the student has to take in order to pass a given degree. In many schools, the student has no choice : he must attend and succeed each of the courses given in the cursus in a given order.
- *n out of m .* Another typical situation is the choice of n items among m (for example, perform 10 exercises out of 100 exercises on unary operators of the relational algebra).
- *Guided multi-level tour.* Another typical situation is the following: prior to pass the exam on SQL, a student must do the exercises on the relational model chapter, then he might choose either to perform the exercises on relational algebra or those on SQL or the other way around. The exercises on relational calculus are optional. He might choose to read the associated course material or not. Note also that successively passing the relational algebra exercises step, implies performing the exercises on unary operators, on binary operators, etc. in a lower level specified guided tour.
- etc.

In order to model e-learning guided tours (egt), we choose hierarchical colour Petri nets [9]. Figure 1 illustrates a Petri net for representing an egt through SQL exercises at the chapter level according to the aforementioned scenario. This egt specifies that any student has to first train through exercises on relational algebra, SQL and relational calculus in any order prior to taking an exam on SQL. With each student is associated a

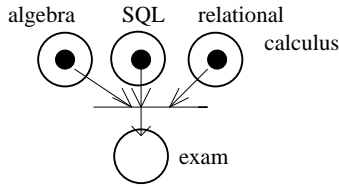


Fig. 1. Egt specification (SQL chapters) (A).

coloured token. The firing of the SQL transition is performed only when the student is in the final place of the lower level Petri net (figure II-B). Thus, the Petri net figure II-B refines the place called SQL in the Petri net figure 1. This Petri net II-B specifies that to pass this step, the student has to perform 3 out of 10 exercises. Note that the firing of an elementary transition (exercise performed) is not specified here. It might be answering the questions of the exercises (the insertion of this event in the student log, see following subsection, might trigger the transition), it might be a stronger condition, such as successfully answering the exercises (the latter process implying a manual or automatic correct answer mechanism with a possible interaction with the teacher, which process when terminated corresponds as well to the insertion of an item in the student log which item triggers the transition).

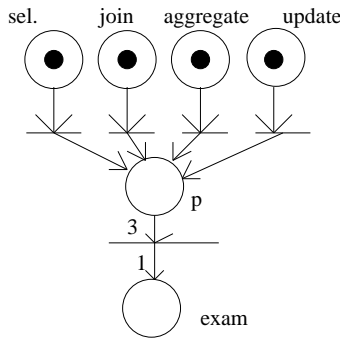


Fig. 2. Egt specification (relational algebra) (B).

Three remarks are noteworthy:

- 1) the specification process of the hierarchical Petri net like egt is very general and only assumes a hierarchical structuring of the document and an egt per level: at each level the egt generation is parametrized by (it takes as an entry) the e-learning material structure (DTD) as well as the teacher specifications.
- 2) It is also important to note that displaying at each level the Petri net with tokens of a given colour or with a counter on the number of tokens in a given place allows for snapshots on the state of progress of a given student, respectively of a set of students.
- 3) the above egt specification is based on the past history of a given student. One might as well specify egt according to other criteria, such as different *a priori* levels of students, multi-pedagogical targets (the same bank of sharable course or exercises documents might be shared among various e-learning situations, actors and targets).

C. Student Log

Each student action is recorded in a log and identified by the log item number. The objective is two-fold: first the egt for a student is dependent on his/her current state (and possibly on his/her past actions) which has then to be recorded. Second our objective is to keep track of all student actions at some level of abstraction for further data mining.

With each student action (basically accessing in read mode a course item, answering in write mode an exercise) corresponds a log item to be inserted into the database. This insertion might trigger a change of state i.e; the firing of a transition in a egt. The actual structure of the database (the log item) depends on the data mining targets as well as on the e-learning situation. However it should include the following attributes:

- item id
- student id
- identification of the piece of document (node or leaf) accessed
- access time and duration
- access mode (read only, write, etc.)
- if the mode is write, answer text
- if the mode is write with answer
 - grade obtained,
 - mistakes types,
- if the mode is write with comment, customized teacher comment according to the answer, etc.

Two remarks are noteworthy.

- 1) Using standard database querying on sequences of log items, one might obtain some aggregated information on grades, duration and higher level nodes accessed (simple extensions of database query standards allow to understand that if a node on selection is accessed, then a node on relational algebra has been accessed). An example of such a query is “get the number of exercises on relational algebra performed by Naomi between day d and day d’, the duration and the sum of grades for these exercises”.
- 2) It is not our purpose to completely specify the information system behind the student log. For example in an alternate design, one could distinguish between student items in the log (one per action) and subsequent teacher items on a given student action such as giving a customized feedback or just a grade. In the latter case, the teacher item should include the student action id (log id) as a foreign key. In that case the firing of a place might be triggered not by the insertion in the log of the student action but by the insertion of the subsequent teacher action.

D. Functional architecture

The three above information components suggest the existence of the following software components: 1) student interface, or e-learning guided tour, for accessing to and navigating through e-learning material; 2) authoring interface for specifying an egt, 3) tutor interface for (a) querying the current state of the system and (b) for data mining. Other components are necessary as well which are not even sketched here. These include a model for exercise answering and methods and code for

response analysis. We end up the section by briefly describing the three above components.

Student interface: The learner interface, or e-learning guided tour (egt), includes a general interface allowing to display the offer and structure of available course material and the interface allowing to access and answer e-learning material according to the egt and the student history. This interface relies on the underlying hierarchical Petri net model for egt.

Authoring interface: The Authoring interface allows specifying egt. Starting from the structure (XML schemas) of the e-learning items the author specifies the constraints on the access to items (interdiction, inclusive or, and, conditions on numbers, etc.) as illustrated on the examples above. A graphical user interface helps the author to specify these constraints which expressiveness is that of the underlying coloured Petri net model.

Data miner and visualizer: Author interface for querying the state of progress and data mining the students works. Standard relational queries on the database (including the student log) allow for snapshots and history of the student(s) progress in the e-learning process. The Petri net underlying modelling allows for identifying individual and group behaviours in the event driven e-learning process. As already suggested, one can visualize and summarize where the students currently stand (in which places they currently are). Keeping track of all student (and teacher actions) allow for subsequent mining as detailed in the last section. It is worth noticing that keeping track of the complete navigation through the e-learning material allows not only the mining of elementary actions but more important of chains of actions in time (complete tours through the e-learning material).

III. PLATFORM DESIGN

We aim at providing an open design to ease contributive developments from both course authors and software developers, allowing teachers and researchers to capitalize and share in an effective manner on e-learning tools and experiences.

E-learning material description: We engineer our Question Answering system with XML technologies. The first motivation is to help authors to describe effectively their e-learning material to the platform (figure 3). XML technologies indeed allow a clean separation between the descriptions of: (a) the exercises structure (i.e. how exercises are organized into questions, how questions are connected with the answers and with the course material), (b) the presentation itself (i.e. how exercises look like on the screen), (c) the answers validation process (i.e. how answers are evaluated), (d) the processing of the XML descriptions themselves (structure filtering and transformations). Moreover many related software tools are currently available and continuously maintained in the open source community, which helps not to reinvent the wheel. The way we are using XML is the following.

Data input validation: The structure and contents of exercises, e-learning guided tours and answers are described in XML in the platform (EGT blocks in the synopsis of figure 3). The use of a schema language for XML, as RELAX NG, allows an automatic validation of the exercises descriptions submitted by authors by matching these descriptions to an XML schema. The same process can apply to student answers, once translated

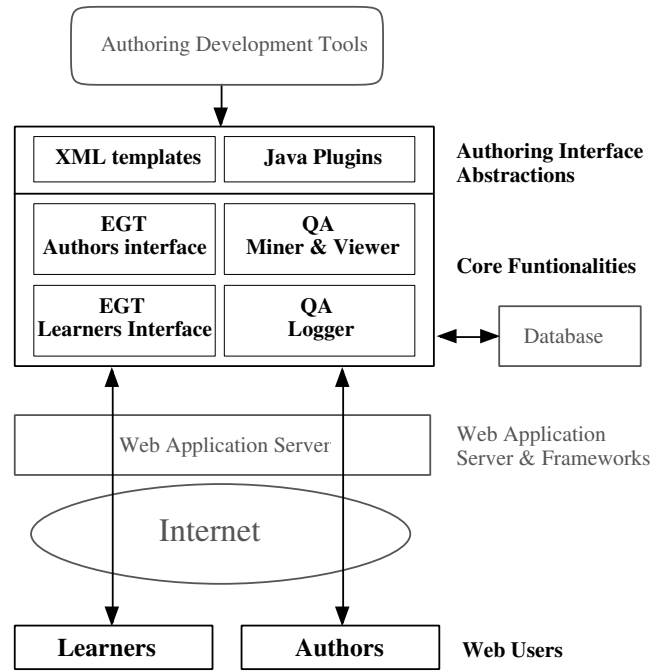


Fig. 3. The LeVinQuam platform architecture overview.

into XML in the platform. The link between exercises and answers can also be made explicit by means of XML syntax. Logical data input validation is a fundamental mean for ensuring a reliable processing of these data.

Authoring interfaces: The platform layer does not pretend to offer to authors an exhaustive set of graphical-oriented authoring tools, which would represent a full development project per se. This is why authoring tools appear at the top, outside the core area in figure 3. In contrast we propose to bridge the gap between authors, developers and the platform by providing an effective set of *authoring interfaces* allowing them to enrich the platform with their own extensions, letting them define new exercises types, new answers validators, and further new graphical-oriented authoring tools to help in these tasks.

In a first phase, authors, with the help of a simple validating XML editor, fill XML templates describing their exercises and answers. A template library providing simple generic types of exercises is proposed to the authors for this purpose. In a second phase graphical-oriented authoring tools extensions to the platform propose to authors a user-friendly environment for designing exercises and output the necessary XML data for the underlying platform authoring interfaces. In both cases, the platform handles the code generation required to put the exercises on line, whatever method has been employed by authors for their description.

Answer validation: Different levels of answer validation can be provided by the platform, according to the nature of exercises, from syntactic validation, like checking whether an answer is a valid calendar date, to complex semantic validations, going through simple semantic validation of closed questionnaires, e.g. checking whether this date is the correct answer. If the question is to solve a mathematical equation, then a simple semantic validation can decide whether the solution is the one expected. However if we want to analyze and validate

more complex formulas, a third level of validation is necessary, that can only be based on software code having knowledge of the application field. To allow authors to provide such validation codes in a robust and effective way a Java plug-in API is supplied by the platform (*Java Plug-in* block in figure 3). For example, in the SQL case study previously mentioned, when a student answer is an SQL query, a Java plug-in validator would send this query to a dedicated SQL server and analyse the server's reply (here, from the standpoint of the platform, Java is a wrapper for SQL). We advocate for the usage of Java within LeVinQam for several reasons. One of them is the huge collection of Java source code available on the internet. Portability is also enhanced with Java, which is a main-stream programming language. Another reason is more technical: the ease of maintenance, compared with scripting languages, and the richness of web frameworks based on J2EE (e.g. the java projects at apache.org).

Data persistence: The last feature of the platform is data persistence. All answers as well as the log of the learner's interaction with the platform are stored in a relational database for further data-mining (*QA Logger* and *QA Miner & Viewer* in figure 3). We use XML transformations to map exercises, answers and guided tours to the relational database schema.

Most of the aforementioned features, especially layers of answer validation and data persistence, have been experimented in a first prototype [5] based on Ganesha [6].

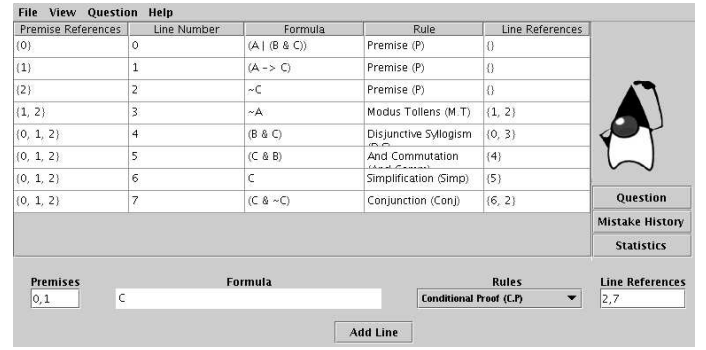
IV. VISUALISATION AND MINING

The rich information model that we adopt for our platform makes it possible to provide facilities to teachers in order to assist them in their pedagogical follow up of students. First, it is possible to convey to teachers (part of the) information that they implicitly have when they do face to face teaching. Second, additional information can be provided due to the technology change. Indeed, having all students (guided) tours, answers, mistakes and, possibly, teachers annotations stored and accessible in a database makes it possible to use queries and Data mining techniques to retrieve pedagogically relevant information for both students and teachers [2]. Here, we focus on the teacher's point of view.

We illustrate our approach with the Logic-ITA, which can be seen as a particular application case of our platform.

The Logic-ITA is a web-based intelligent teaching assistant system for the domain of formal proofs in propositional logic currently in use at the Information Technologies School of the University of Sydney. It provides an environment where students can practice formal proofs of logic at their own discretion, receiving step-by-step, contextualised feedback. They can choose to create new exercises, select exercises in the exercise database, or ask the system for one adapted to their needs. The system stores, for each student, every step entered, along with any mistake the student may have made and collates all this information into a database. This makes the information model of the Logic-ITA quite close to the one proposed in our platform. LeVinQam generalizes the Logic-ITA by providing a model of learning tours.

We need to explain the structure of an exercise to make the following clearer, the reader may refer to [1] for more details.



Premise References	Line Number	Formula	Rule	Line References
(0)	0	$(A \vee (B \wedge C))$	Premise (P)	()
(1)	1	$(A \rightarrow C)$	Premise (P)	()
(2)	2	$\neg C$	Premise (P)	()
(1, 2)	3	$\neg A$	Modus Tollens (M.T)	(1, 2)
(0, 1, 2)	4	$(B \wedge C)$	Disjunctive Syllogism	(0, 3)
(0, 1, 2)	5	$(C \wedge B)$	And Commutation	(4)
(0, 1, 2)	6	C	Simplification (Simp)	(5)
(0, 1, 2)	7	$(C \wedge \neg C)$	Conjunction (Conj)	(6, 2)

Premises	Formula	Rules	Line References
0,1	<input type="text" value="C"/>	Conditional Proof (C.P)	2,7

Fig. 4. Screenshot during an exercise.

Exercises start with a given set of premises, i.e. a set of well-formed formulae (wff) of propositional logic, and exactly one wff, the conclusion. The task then consists of deriving the conclusion from the premises, step-by-step, using laws of equivalence and rules of inference (we will refer to both of these as rules for the rest of this paper). Figure 4 shows a screen shot of the interface. Here the student was given the first two lines (lines 0 and 1) and the conclusion at the bottom left corner, i.e. C . For each step, the student must fill out a new line, entered at the bottom of the screen. The student needs to do the following:

- enter a formula in the *Formula* section,
- choose, from a pop-up menu, the rule used to derive this formula from one or more previous line(s) (*Rules*),
- the references of those previous lines (*Line References*) and
- the premises the formula relies on (*Premises*).

For example in Figure 4, the student is currently deriving the formula C , using the rule *Indirect Proof* and the formulae of lines 2 and 7. Because lines 2 and 7 rely respectively on premises 2 and 0,1,2 (as can be seen in the first column of the screen) and *Indirect proof* removes the premise 2, the line entered therefore relies on premises 0,1. It is actually the last step of this exercise, deriving the conclusion.

At each step, the system checks the validity of the data entered by the student. There are different types of mistakes, and each of them is labelled with a meaningful title for the teacher. For example, the mistake message *Wrong reference lines* means that the student has not provided the right lines of reference the rule applies to.

A. Retrieving implicit information provided by face to face teaching

In face to face teaching, teachers would be aware of students who succeed completing exercises and students who fail, on how students use the tool, in a thoughtful manner or just trying any possible exercise, any possible rule one after the other – at least as far as classes are not too big. Experiments with the Logic-ITA indicate that this kind of information on students' behaviours can be conveyed to teachers.

The aim of the tool is to help students grasp formal proofs. In the case students make mistakes but finish successfully exercises, teachers do not need to worry. Teachers need to be aware of students not completing successfully exercises since they may have difficulties. In order to characterize these latter students, a k-means clustering [8] has been applied, taking

into account the recorded mistakes. The clustering yields three classes. Class 1 is composed of students making few mistakes, class 2 of students making an intermediate number of mistakes and class 3 students making many mistakes, see [3] for more details. Then several graphs have been produced. A first graph plots *logins* (i.e. student identification) against *exercise-id* (exercise identification). Thus this graph visualizes the various exercises attempted by each student. The trend given by this graph is that students of class 1 attempt more exercises than students from class 2 or 3. A second graph plots *logins* against *mistake-messages*. The trend given by this graph is that students from class 2 or 3 make more different kinds of mistakes than students from class 1. Students from class 1 make the mistakes that are most usually made by everybody using the tool. Plotting *logins* against *logical-rules* used in the non-completed exercises gave the graph given in Figure 5. This graph shows vertical lines for several students from class 2 or 3 only, not from class 1. Students from class 1 constitute the narrow green strip in the middle of the graph. A vertical line means that all rules have been tried while doing the exercises. They suggest that these students have just tried one rule after the other from the pop-up menu, apparently adopting a behaviour of "guess and test" strategy. Awareness of these behaviours may lead teachers to differentiate their pedagogy.

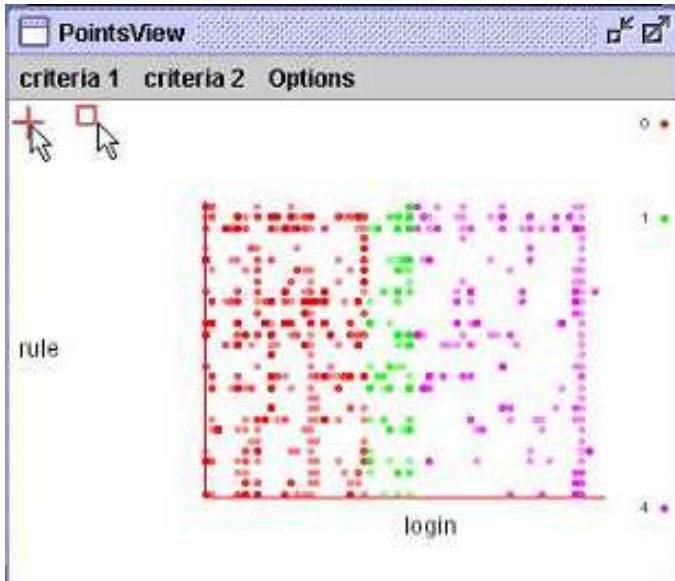


Fig. 5. Plot of Students against Logic Rules.

B. Extracting hidden information

Using Data Mining techniques on students' answers stored in the database can lead to discover patterns that quite often remain hidden or not well defined otherwise. We have used the association rule algorithm to all the answers from the Logic-ITA to be aware of mistakes often made together while solving exercises. Figure 6 shows parts of the result, [10] gives more details.

As an example, the association *Rule can be applied, but deduction incorrect* \rightarrow *Premise set incorrect* means that while solving an exercise, if a student makes the mistake *Rule can*

association	supp.	conf.
<i>Rule can be applied, but deduction incorrect</i> \rightarrow <i>Premise set incorrect</i>	61%	82%
<i>Premise set incorrect</i> \rightarrow <i>Wrong number of line references given</i>	67%	87%
<i>Rule can be applied, but deduction incorrect</i> \rightarrow <i>Wrong number of line references given</i>	65%	87%

Fig. 6. Associations found focusing on mistake messages.

be applied, but deduction incorrect then s/he makes also the mistake *Premise set incorrect*, this association has a support of 60%. and a confidence of 82%. Support makes sure that only mistakes occurring often enough in the data will be taken into account. Confidence is a measure of how much *Y* is really implied by *X* in the rule $X \rightarrow Y$.

First, we explain what these mistake messages mean, referring to the example shown in Figure 4. Consider line 3. If the student gives the formula *A* instead of $\neg A$, the mistake *Rule can be applied, but deduction incorrect* is made. Indeed, *Disjunctive Syllogism* can be applied, but the negated left side of the formula given line 1 can be deduced, as shown in Figure 4, not the positive form as written here. Suppose now that the student gives only 1 in the *Prem.* field. Then the mistake *Premise set incorrect* is made. Finally, suppose that the student gives only 1 in the *Refs.* field. Then a *Wrong number of line references given* mistake is made, because 2 lines of reference are needed.

The associations found show relations between mistakes involving line numbers in the premises (*Premise set incorrect*), line numbers in the reference lines a logic rule applies to (*Wrong number of line references given*) and incorrect use of logic rules (*Rule can be applied, but deduction incorrect*). This confirms what human tutors had sensed. First, students often have difficulties at grasping all details required in a proof: one has to provide not only a logic rule, but also the lines it applies to, and these are different from the premises involved. Second, students do not realize at once that there are two kinds of logic rules: rules of equivalence that are applied to one formula only, and rules of inference that are mostly applied to two formulas. Most importantly, rules of equivalence can be applied to subparts of a formula whereas rules of inference can only be applied to whole formulae. For example in the formula $((A \wedge B) \rightarrow C)$ we can validly replace $(A \wedge B)$ with $(B \wedge A)$ in the formula using the rule of equivalence *And Commutation* but it is not valid to deduce *B* from $((A \rightarrow B) \rightarrow C)$ and *A* using the rule of inference *Modus Ponens*. Following these findings, presentation of the course material has been revised to put more emphasis on the differences between rules of equivalence and rules of inference.

REFERENCES

- [1] D. Abraham, L. Crawford, L. Lesta, A. Merceron and K. Yacef, "The Logic Tutor: A Multimedia Presentation", *Interactive Multimedia Electronic Journal of Computer-Enhanced learning*, Vol. 3, Nb. 2, Nov. 2001.
- [2] B. Aguado, A. Merceron and A. Voisard, "Extracting Information from Structured Exercises", *Proceedings of the 4th International Conference on Information Technology Based Higher Education and Training ITHET03*, Marrakech, Morocco, (2003).

- [3] Benchaffai M., Debord G., Merceron A., and Yacef K., *TADA-Ed, a tool to visualize and mine students' work*. Submitted paper. 2004
- [4] P. Brusilovsky, "Adaptive and Intelligent Technologies for Web-based Education" *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching*, 4, 19-25, 1999
- [5] Laurent Wargon, "Une plateforme d'apprentissage de SQL en ligne", *Rapport interne*, ESILV/GI, January 2004.
- [6] <http://www.anemalab.org/ganesha/>, Ganesha, a Learning Management System
- [7] M. Grandbastien, L. Oubahssi, G. Claes, "A process oriented approach for modelling on line Learning Environments", *Supplementary Proceedings of Artificial Intelligence in Education AIED2003*, University of Sydney, Australia, pp. 140-152, July 2003.
- [8] Han J., and Kamber M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2001
- [9] K. Jensen "An Introduction to the Theoretical Aspects of Coloured Petri Nets." *A Decade of Concurrency*, J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.), Lecture Notes in Computer Science vol. 803, Springer-Verlag, 230-272, 1994
- [10] A. Merceron and K. Yacef, "A Web-Based Tutoring Tool with Mining Facilities to Improve Learning and Teaching", *Proceedings of the 11th International Conference on Artificial Intelligence in Education*, Sydney, Australia, pp.201-208, 2003