The Logic-ITA in the classroom: a medium scale experiment

Kalina Yacef

The Logic-ITA in the classroom: a medium scale experiment

Kalina Yacef. *School of Information Technologies, University of Sydney, NSW 2006, Australia*
kalina@it.usyd.edu.au

Abstract. This paper presents the experiment and consequent evaluation of introducing the Logic-ITA in a second year tertiary undergraduate class. The Logic-ITA is a web-based Intelligent Teaching Assistant system, aimed at alleviating some of the problems caused by large classes or distance learning. Its domain of application is the construction of formal proofs in logic. The system acts as an intermediary between teacher and students: on one hand, it provides students with an environment to practice formal proofs with feedback and on the other hand it allows teachers to monitor the class's progress and mistakes. It is complementary in the sense that it does not aim to replace any of the existing interactions between teachers and students. Since its introduction, over 600 students have used the tool. Evaluation shows a significant improvement in students' assessment results with an effect size of around 1 sigma.

Keywords. Intelligent teaching assistant, Evaluation of ITS, Student data analysis, web-based ITS

INTRODUCTION

The AIED community has devoted a lot of work over the past four decades to inventing principles and tools to assist learners, in particular through individualised learning. Whereas learners have generally been the focus, much less attention has been paid to teachers, who are generally perceived as the administrative managers of the tutoring system and/or one of the designers of the system. The approach followed in the work presented here is slightly different. Rather than solely helping learners, our approach is to also help teachers to teach better and more efficiently. This is particularly needed when teachers are a scarce resource, such as in classes with large numbers of students or in distance education. The pedagogical expertise and the face-to-face interactions they provide are invaluable but there are not enough of them. In this sense, our work is part of the growing interest in the teacher's role and his/her integration as a target user in AIED (Jean 2000; Kinshuk 2001; Leroux, Vivet & Brezillon 1996; Van Labeke 1999; Virvou & Moundridou 2001; Vivet 1992; Yacef 1999; Yacef 2002).

Studies show that students and teachers experience problems induced by large classes (Habeshaw, Gibbs & Habshaw 1992) and by web-based teaching (Hara & Kling 1999; Pittinsky 2003). Some of the problems identified in the literature are the students' frustration of feeling lost in the mass, of not receiving appropriate and timely feedback on their work, of feeling isolated. On the teachers' side, time consuming tasks, poor visibility of students' progress and problems seem to be dominant. The Logic-ITA is an experiment to try to reinforce the fragile and crucial

relationship between teachers and students in the context of online or large class teaching and bring them a lot closer together, modelling some aspects of the valuable direct contact that can occur when the ratio between teachers/ facilitators and students is low.

The Logic-ITA is a web-based Intelligent Teaching Assistant (ITA) system. ITAs are dedicated both to learners and teachers (Yacef 2002). Their purpose is to support the educational or training process in an intelligent way by assisting the teacher in his/her tasks as well as helping learners to learn. They can take a significant load off the teachers, assist them in tedious or complex tasks, keep track of the students' results, report problems whilst helping learners to practice at their own pace in an adapted environment, receiving feedback and tailored exercises. Assisting the whole learning process and treating the teacher as a target user rather than replacing him/her is the key philosophy of an ITA. Teachers remain in control of the teaching and are supported by the ITA.

We wanted to conduct an experiment on a small domain to investigate the benefits an ITA could bring to large classroom teaching. The domain of application of the Logic-ITA is formal proofs of propositional logic. The tool was introduced in 2001 at the University of Sydney in an undergraduate course. Students in 2001, 2002 and 2003 significantly outperformed students in 2000 on various tests that required them to perform logic proofs.

This paper is organised as follows. In the next section we describe the Logic-ITA: after a summary of its history and the overview of the system, we explain each of its components in turn. In the following section we present the experiment, describing the student and teacher population as well as the context in which the system was used. We then report and discuss the results of the quantitative and qualitative evaluations, before presenting our conclusion.

THE LOGIC-ITA

History of the Logic-ITA

The research motivation behind the creation of the Logic-ITA was to discover ways to compensate for the low amount of interactions between teachers and students in large classes. Propositional logic was a good candidate for applying our concept: whilst formal proofs can be considered amusing once the concept is grasped, reaching that stage can take time and many trials. Practice is the key to understanding the concept of formal proofs, and answers must be checked by an expert, such as the lecturer or the tutor[1]. However checking and marking them are quite tedious and very error prone for a human and, unfortunately, the face-to-face contact is not what it should be, due to the high volume of students.

Formal logic proofs are taught in a computer science theory course (2nd year undergraduate) covering formal languages and logic. In 2000, the lecturer in charge at the time and I, who was about to take over, had observed that many students make common mistakes; hence tutors and lecturer found themselves repeating the same explanations to every student. The lack of time and opportunity was making this task very inefficient. Automating some of the teaching team's

[1] In Australia, students have lectures and small problem-solving sessions called tutorials, in groups of up to 20. The person who facilitates these tutorials is called a tutor.

expertise and making it accessible to every student at anytime was the motivation behind the Logic Tutor (Abraham, Crawford, Lesta, Merceron & Yacef 2001).

With the Logic Tutor, students had access to a substitute tutor, but the teacher had no idea how the class was progressing. The other direction was missing. One way to partly rectify this was to follow a teaching assistant model, where the lecturer is in charge of the curriculum decisions, relies on teaching assistants to help students and also receives feedback from them about progress, problems and so on.

This is what led to the Logic-ITA. Besides the addition of a student model component and delivery of tailored exercises, the Logic-ITA comprises the LT-Configurator, a teacher version of the Logic Tutor, with authoring functionalities such as the configuration of the exercise levels and the progression of students through these levels, and the LT-Analyser, a database of all student models, which the teacher can query to find out about the class's results, progress and problems (Lesta & Yacef 2002).

Its web-based version was released soon after (Abraham & Yacef 2002), as the Unix version was too constraining for the users. The Logic-ITA obviously requires regular access to the student models in order to keep the database of the LT-Analyser up to date. In 2001, the only way to centralise the student models in order to analyse their data was to ask students to use a server version under their Unix undergraduate account. We used this method during the teaching weeks and released the independent Windows version during the exam study week. The web version solved these inconveniences. It brought geographical and time freedom for the students and the indispensable centralisation of the student models.

Overview

The role of the Logic-ITA is to help students practice as much as possible with feedback and report to the teacher about the class's stages of learning, problems and so on. Figure 1 illustrates the various processes involved.

The teacher (right) first sets up the various parameters in the Logic-ITA to define the high-level curriculum, using the LT-Configurator. S/he can set up, for example, the number and characteristics of the teaching levels and the criteria for progressing through these levels. S/he can also add (or delete) exercises to the exercise database. As a result, the exercise database will be indexed using these criteria.

Then the students use the Logic Tutor. They can either practice on exercises that are tailored to their needs and suggested by the system, select any existing exercise or create their own exercise. If they choose a tailored exercise, the system will use the high-level curriculum criteria combined with the student's history to define training goals. For example, the high level curriculum, which is defined by the teacher and common to all students, may recommend exercises using rules A, B or C and a difficulty level *d*. The current student has used rule A with no mistake, has a history of problems using rule B and never used rule C. The exercise retrieved will therefore be more likely to involve the use of rules B and C with a level of difficulty *d*.

For each student, the history of usage, along with various statistics and a copy of all exercises attempted is saved in their respective student model. The LT-Analyser then constructs or updates a database collating all of the student models. The teacher can query this database to find out how the class is performing, where the main difficulties are and so on.

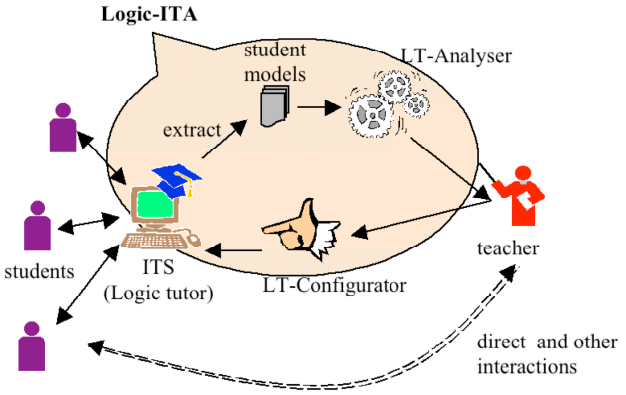


Fig. 1. Logic-ITA's role as an intermediary between teacher and students.

The teacher can then use this information to adapt the content of his/her lectures and interactions with the students. S/he may also discover individuals who are not progressing at all and decide to intervene.

Description of the components

The Logic-ITA comprises of three components: the Logic Tutor, the LT-Configurator and the LT-Analyser. Let us explain each of these in turn.

Logic Tutor

The Logic Tutor is a Java-based Intelligent Tutoring System (see (Abraham, Crawford, Lesta, Merceron & Yacef 2001) for an online multimedia article). It allows students to build formal proofs in propositional logic whilst receiving step-by-step, contextualised feedback. We should state that there are already many computer-assisted educational systems for this domain. For example DEEP THOUGHT (Croy 1999; Croy 1989; Scheines & Sieg 1994) or the CPT (Scheines & Sieg 1994) are two systems with different styles of interfaces. Studies suggest that interfaces supporting backward construction of proofs as well as forward construction are more efficient. In the Logic Tutor, we kept the same style as the one used in previous years for the course, i.e. a conventional one allowing only forward construction of proofs. The reason behind this is that our aim was to design an Intelligent Teaching Assistant system and then evaluate its usefulness using the previous year as a control group. It would have been more difficult to interpret results if we had changed the style of the interface.

There are often many ways to prove an argument valid. The important aspect is that the reasoning must be sound. The actual path followed is not important, as long as each step is valid. In this regard, our approach is less sophisticated than one such as Model-Tracing (Anderson, Corbett, Koedinger & Pelletier 1995), which identifies the student's reasoning by checking his/her answers against predetermined solutions. The Logic Tutor instead assesses the *validity* of each step on the fly, but not its *appropriateness*. During the exercise, the system can only assess whether the line entered by the student is logically valid, and whether or not the conclusion was

reached but does not know how far the conclusion is. However, once the exercise is finished, it can then evaluate whether all the steps were actually useful. This has two consequences on the evaluation of the student and on the feedback provided. The feedback provided to the student is only related to the validity of the current step, not whether the step is moving towards the solution or not. For example, the Logic Tutor would not give a feedback such as "the step you entered is valid but it is not going to be of any use in reaching the conclusion". This means students have total freedom in the reasoning they choose to follow. The performance of a student is calculated in terms of whether or not the conclusion was finally reached, whether or not mistakes were made along the way and whether or not useless steps were entered.

In a nutshell, the underlying principles in the Logic Tutor are that:

- immediate feedback facilitates the learning process (Anderson, Corbett, Koedinger & Pelletier 1995; Mark & Greer 1995).
- scrutinising one's student model enhances learning (Kay 2000). Students, at any time, can consult the data saved in their student model, reflecting on their past mistakes and consult general statistics about their results.
- students are in control of what they want to practice. Whilst the system can suggest exercises that are adapted to their needs, students can always choose to create their own exercise (that they may have taken from a book for example), or select an exercise in the database that will make them practice a particular rule of their choice.
- the curriculum is based on MOST, a model that is described later in the paper.

Interface and system behaviour. Figure 2 shows a screen shot of the interface. In an exercise, the student is given a set of premises, i.e. a set of well-formed formulae (wff) of propositional logic, and exactly one wff (the conclusion). The set of premises may be empty, in which case the conclusion, if proven, is a tautology. The exercise 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). For each step, the student must enter a formula, choose, from a pre-defined pop-up menu, the rule used to derive this formula from one or more previous line(s), the references of those previous lines, and the premises the formula relies on. For example in Figure 2, at line 4, the student would have derived the formula (B&C), using the rule "*Disjunctive Syllogism*" using the formulae of lines 0 and 3. Because lines 0 and 3 rely respectively on premises {0} and {1,2} (as can be seen in the first column of the screen), line 4 therefore relies on premises {0,1,2}.

After each line is entered, the system checks the validity of the formula entered, following a principle of cascading mistakes. It first checks its syntactical validity (right type of data, syntax of the formula) then its logical validity. In the latter, the formula, the rule, the references and premises entered must all be consistent. If not, then the system checks whether altering just one could be valid. If this is the case, the system will use this substitution to provide hints to the student. For example, suppose the line entered is invalid as such but by changing the rule from *Modus Ponens* to *Modus Tollens* the line becomes valid. The hint would then be to try applying *Modus Tollens*. If this is not the case, then the system looks up the database of common mistakes and tries to match the line with a common mistake. As these come with remedial hints, the student receives the corresponding feedback. For example a common mistake is to apply *Simplification* before *Commutation* to derive C from (B&C). The common mistake has an associated feedback and the student reads "This was an invalid application of Simplification

(Simp). Applying *Simplification* to (B&C) only lets you deduce the left hand side: B. Try to use *Commutation* first".

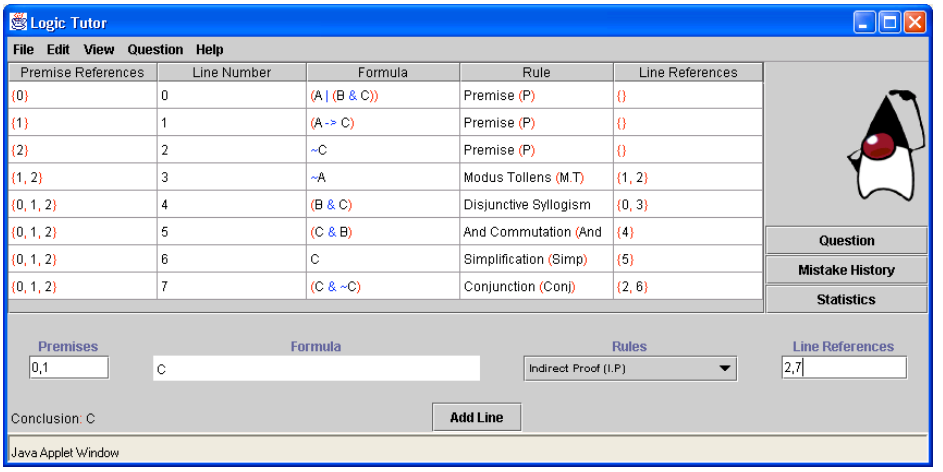


Fig. 2. Screen shot during an exercise.

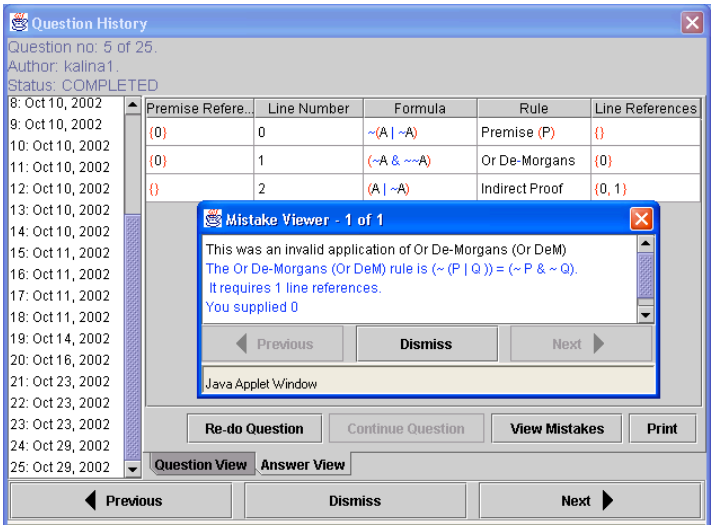


Fig. 3. Browsing of mistakes in past exercises.

Architecture of the Logic Tutor. The Logic Tutor has a classical internal architecture involving an expert module, a pedagogical module, student models and interface. The expert module contains the expertise of the system in logic. It assesses each step on the fly and produces corresponding feedback when errors occur. The pedagogical module is the system's engine. It contains the high-level rules for sequencing exercise training objectives according to the student model data and then suggests appropriate exercises that the student is free to choose.

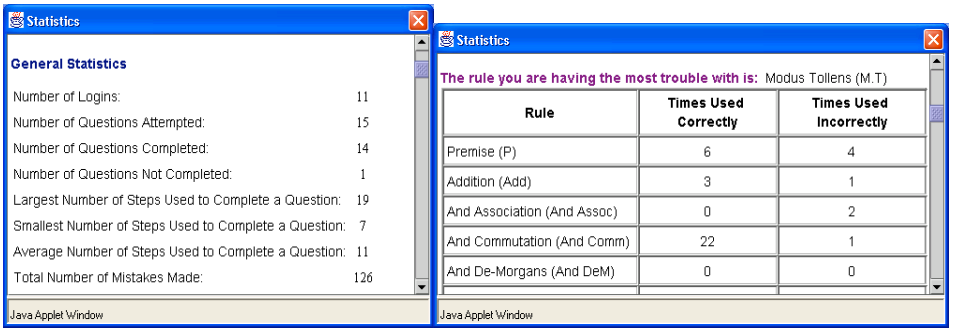


Fig. 4 a and b. Two screen shots of the statistics window.

We will now describe the student model data, as it forms the input to the LT-Analyser.

Student model data. The Logic Tutor creates and maintains a student model for each user. In effect, it is a mix of overlay and error models, but it actually records all the attempted exercises, along with all the mistakes made. The student can browse through their past questions and view their past mistakes (via the right bottom button on Figure 2, or via the *Question* menu). A sample of screen shot is show below in Figure 3. The student model also stores the student's current level, the performance obtained for each exercise and basic student information.

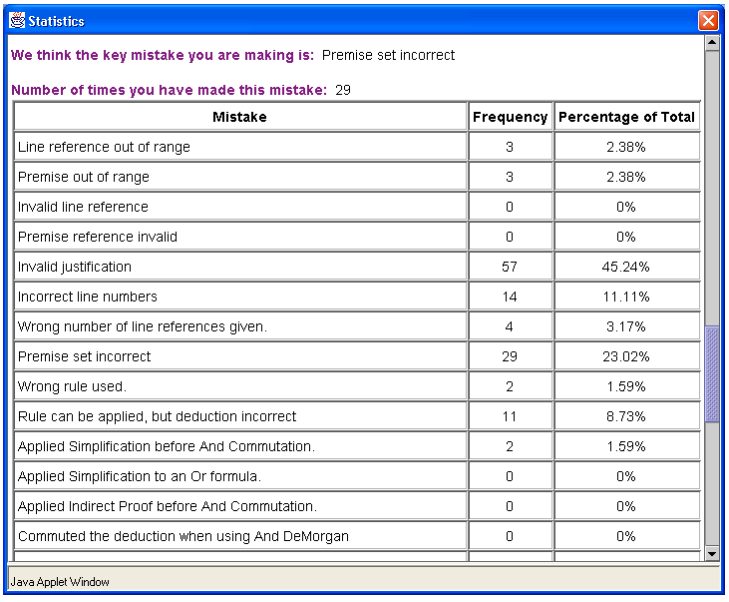


Fig. 5. A third screen shot of the statistics window.

Via the Statistics menu, the student can also access general statistics about his/her usage of the Logic Tutor. This is illustrated in Figure 4 and Figure 5.

The statistics window also tells the student which was the hardest question completed, and which question gave him/her the most difficulty.

LT-Configurator

The LT-Configurator is an administrative version of the Logic Tutor comprising of a curriculum sequencing authoring tool and an exercise management tool. The high-level curriculum is defined through sets and groups of rules, levels of difficulties, and criteria for students to progress from one level to the next (following the model MOST which is defined in the next section). This level is used to classify exercises and will be used by the Logic Tutor to select tailored exercises, with finer grain of training objectives being decided on–the-fly. The Logic Tutor first delimits a set of exercises matching the level, the number of steps and the potential rules. This set contains only exercises not yet attempted by the student. Each of these exercises is given a weighting, equal to the average weight of the rules in the exercise. The rules are given weights according to the student model data. The weight of a rule is determined by the ratio of correct to incorrect uses of that rule by the particular student. The formula used to calculate this is as follows:

Rule weighting = (TUI – TUC)
(TUC is equal to the number of times the rule has been used correctly and
TUI is equal to the number of times the rule has been used incorrectly)

Where a rule has never been used, it is arbitrarily assigned a weight of 3. This means that the Logic Tutor will try to select exercises which require the unused rule in preference to exercises which require rules that the user has not repeatedly used incorrectly ("Repeatedly" here means incorrectly using the rule at least 3 more times than the number of correct usages of that rule.) However, if a rule has been used incorrectly a significant number of times, then the Logic Tutor will give preference to questions containing that rule over questions that contain an unused rule. The exercise obtaining the higher average weight will be selected for the student. For example, if a student is at level 2 and has made repeated mistakes with, say, the *Modus Tollens* and *Addition* rules, an exercise using these two rules is more likely to be selected next.

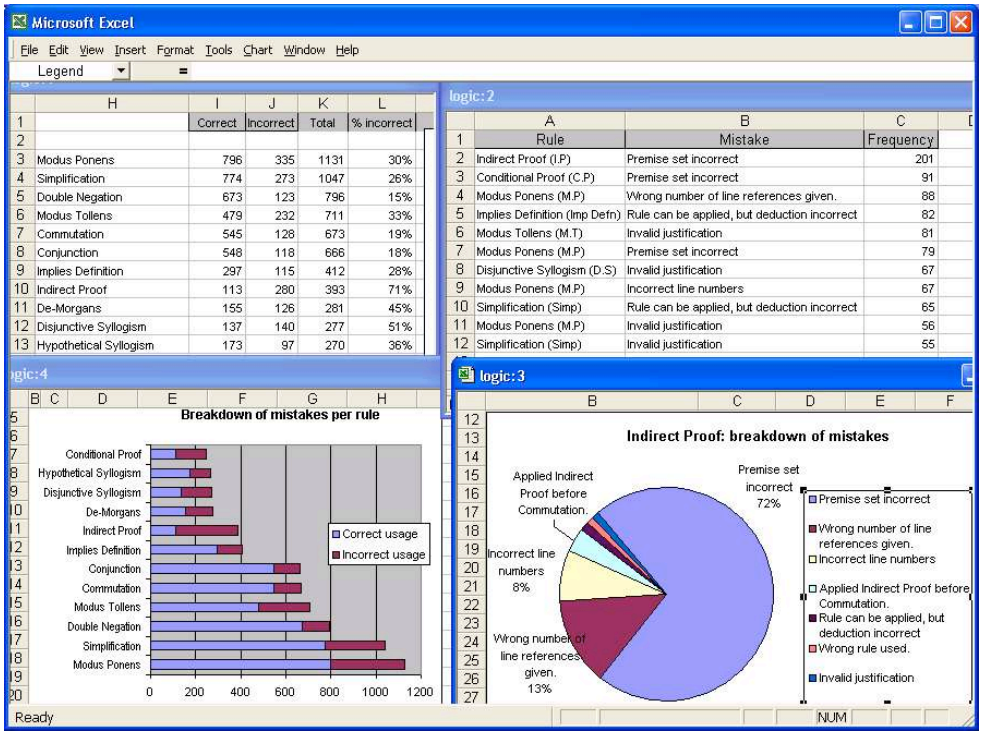
Naturally, students always have the freedom to do any exercise they like, regardless of the difficulty.

LT-Analyser

The LT-Analyser collates all the student information in a database that the teacher can query and visualise graphically. The student models, all centralized in one place on the server, contain the history of all exercises attempted and mistakes made. The LT-Analyser scans all the student models and builds a database collating all that information. Basically the database contains a table with the details of each mistake made for each question for each student, a table with level information for each question, a table containing each student's performance for each question attempted, a table with the rules used correctly by each student, a table with the count of logins and another with the student levels.

The database is in Microsoft Access and is connected to Microsoft Excel. The teacher has then the choice of querying the database with either software and to visualise graphics in MS Excel. Examples of queries and charts are shown in Figure 6: breakdown per rule of incorrect

and correct usage (top left window), associated bar graph showing the proportion of mistakes per rule (bottom left window), breakdown of type of mistakes for each rule (right top window), and breakdown of mistakes for one particular rule (right bottom chart, for the rule *Indirect Proof*). Other common queries are the exercises producing the most mistakes, or the average performance for a particular exercise.



	H	I	J	K	L
1		Correct	Incorrect	Total	% incorrect
2					
3	Modus Ponens	796	335	1131	30%
4	Simplification	774	273	1047	26%
5	Double Negation	673	123	796	15%
6	Modus Tollens	479	232	711	33%
7	Commutation	545	128	673	19%
8	Conjunction	548	118	666	18%
9	Implies Definition	297	115	412	28%
10	Indirect Proof	113	280	393	71%
11	De-Morgans	155	126	281	45%
12	Disjunctive Syllogism	137	140	277	51%
13	Hypothetical Syllogism	173	97	270	36%

	A	B	C
1	Rule	Mistake	Frequency
2	Indirect Proof (I.P)	Premise set incorrect	201
3	Conditional Proof (C.P)	Premise set incorrect	91
4	Modus Ponens (M.P)	Wrong number of line references given.	88
5	Implies Definition (Imp Defn)	Rule can be applied, but deduction incorrect	82
6	Modus Tollens (M.T)	Invalid justification	81
7	Modus Ponens (M.P)	Premise set incorrect	79
8	Disjunctive Syllogism (D.S)	Invalid justification	67
9	Modus Ponens (M.P)	Incorrect line numbers	67
10	Simplification (Simp)	Rule can be applied, but deduction incorrect	65
11	Modus Ponens (M.P)	Invalid justification	56
12	Simplification (Simp)	Invalid justification	55

A word about the underlying model MOST

The underlying model of learning in the Logic-ITA is based on MOST, a Model for Operational Skill Training (Yacef 1999). It was previously used in the prototype of an ITA for Air Traffic Control training. Cognitive Science provides a number of theories each giving an insight into a particular aspect of skill acquisition and utilisation. Some focus on how a particular skill is developed, others on the range of skills exhibited in human performance, yet others on what characterises expert behaviour or on how expertise takes place and so on. MOST takes into account major cognitive theories and suggests simple principles to regulate the training objectives. In short, MOST regulates the type of situations (exercises or part of exercises) that the learner faces and adapts the workload to the current level of skill acquisition. These two elements compose the context in which learners will practice their skills. MOST recognizes two important steps in the learning process: the creation of situation patterns and the development of automation.

- *Situation patterns*: Sweller (Sweller 1993) and Boy (Boy 1991) draw attention to the notion of *situation*, which is very important in the development of automaticity. Situation patterns (Boy 1991), also called patterns (Rasmussen 1983) or schemes (Sweller 1993), are situational structures composed of highly compiled knowledge and are stored in the long-term memory. They are the result of a long period of learning on similar situations, and are organised towards a goal. Situation patterns become more complex, dynamic and numerous as expertise develops. They reduce the amount of information dealt with in short-term memory and allow faster processing of information.
- *Automation*: Fitts and Posner (Fitts & Posner 1967) and Anderson (Anderson 1983) distinguish several learning phases for each skill: the cognitive, associative and autonomous phases (or declarative, knowledge compilation, procedural in Anderson's model). These models explain, in a detailed way, the automation of a skill (ie the attainment of the last phase). Whilst these models were designed for skills that can be automated (such as making additions), the same principles can be applied to those more complex and risky that cannot be fully automated (such as air traffic control, or driving a car).

According to MOST, the learner must first practice in various situations that require the particular skill. This practice must first be carried out at a slow rate so that the learner can build appropriate situation patterns without being overwhelmed. When the performance measures indicate that the learner is doing well, and is fast and efficient, it is assumed that the step of building adequate situational patterns is reached. The workload then increases progressively, often by introducing other tasks, requiring different skills, so as to challenge the learner without overwhelming him/her. This is the stage where automation develops. The performance measures give an indication whether automation is taking place: a constant good level of performance is a sign that it is occurring. If there is a significant drop in performance, MOST reduces the workload.

MOST was primarily designed for operational skills, which are defined as the use of declarative and procedural knowledge in a timely and efficient manner to meet a particular goal or a set of constraints. They are the skills that allow people to deal with unseen situations and problems. Naturally, the domain of formal proofs is much simpler and better defined than that of Air Traffic Control. But it retains the facts that (i) students are not required to automate the conduction of formal proofs and (ii) they need to assimilate the underlying process of conducting proofs as opposed to learning about specific rules. In fact the rules themselves do not matter that much and if they were changed for another set of complete rules, the student should still be able to conduct the proof. A much simpler version of MOST is integrated in the Logic-ITA. Skills and tasks have been reduced to the knowledge of how to apply rules and the ability of solving exercises using minimal steps. As we will see, situations are described here in terms of the set of rules used in the proof and the workload is defined in terms of number of steps and the variety and complexity of rules required in the proof.

EXPERIMENT

The aim of this experiment was to investigate whether an intelligent teaching assistant could help teachers teach better and learners learn better. We will describe how teacher and students used each component of the Logic-ITA and present the findings of our evaluation.

Population and description of the teaching context

The Logic-ITA was introduced in class in 2001, in an undergraduate course entitled 'Languages and Logic' at the School of Information Technologies, University of Sydney. The course familiarizes 2nd year undergraduate students with important theoretical models underlying computer science: formal languages (automata, grammars, parsing) and then propositional and predicate logic. It is one of the most theoretical undergraduate courses offered in the school and unfortunately many of our computer science students struggle with theory. Hence practice is not only useful but also fundamental for them to grasp the concepts. The course includes programming assignments and practical exercises. However the timeframe is quite short: 2 hours of lectures and 1 hour of tutorial per week, over 13 weeks. The actual logic part is taught over 5 weeks, i.e. 10 hours of lectures and 5 hours of tutorial plus homework. Formal proofs require practice to reach a stage where their mechanism is well understood. In the past, we observed that the face-to-face contact time was not sufficient to provide students with good and sufficient feedback. Only self-disciplined students sufficiently prepared their homework for their tutorial and received the feedback they needed. As of 2001, the Logic-ITA was particularly useful to the students as it allowed them to practice on their own and to receive immediate feedback. It can be assumed that the same proportion of students prepared themselves for the tutorials, but those who did not still had the chance to do so before the exam.

The control group is the 2000 class, who did not use the Logic Tutor. Students had the same amount of contact times with their lecturer and tutor, and followed the same curriculum for that unit for the parts that we assessed. 431 students sat the final exam.

In the 2001 class, 390 students sat the final exam. They used a Unix version of the Logic Tutor during teaching time (and not the Windows version because then student data would be stored locally and using the LT-Analyser would be impossible) but we released the standalone Windows version during exam preparation time.

In the 2002 and 2003 classes, respectively 245 students[2] and 132 students[3] sat the final exam. These students used the web-based Logic Tutor, in a more extensive manner than the previous year, because of an extra assignment on the Logic Tutor from 2002 and probably also because of the more convenient aspect of a web-based system.

In terms of prior knowledge, the prerequisites for the course remained the same throughout these four years: students had to have completed a discrete maths course or equivalent and a first year programming course.

Finally, there was a change of lecturer from 2001. I personally took over the course in 2001. However, I was partially involved in 2000 and, apart from the Logic Tutor, I used similar

[2] The drop in numbers in 2002 is due to the fact that the School curriculum was redesigned and more courses were offered to second year students in 2002. This can also mean that students who were enrolled in the course were on average more motivated, given that they had the choice.

[3] The additional drop in 2003 was due to the general and worldwide downturn in numbers of students undertaking tertiary IT studies.

teaching methods and materials as my predecessor, who was involved in the early stage of the Logic Tutor and with whom I stayed in close contact.

Usage of the Logic-ITA

LT- Configurator

In effect, the curriculum we designed recommended that students first practice on short exercises involving laws of equivalence *or* simple rules of inference (ie not *Indirect Proof* (IP) or *Conditional Proof* (CP), which both involve addition and deletion of premises), without yet mixing the two categories; then on a mix of these two sets; and finally on any rule or law including IP and CP. Throughout these stages, a pacing of the difficulty is made by setting limits on the length of the exercises (ie the number of lines in a possible solution) and the number of rules they involve. The longer the proof needs to be, the more vision is required. This pacing is based on MOST, where the situations are described in terms of rules involved in the proof, and where the workload is related to the variety of these rules and the number of steps involved in the proof.

The reason behind the initial separation of equivalence laws and inference rules is that they are of different types and uses. The former come from logical equivalences, therefore can be used to substitute any subpart of a wff (for example, in the wff ((A&B)→C), one can derive ((B&A)→C) using the law of *commutativity*). Rules of inference come from logical inferences. They should only be applied to a whole formula (for example, the *Simplification* rule says that one can derive A from (A&B), but, in the wff ((A&B)→C), it is not logically valid to derive (A→C)). Hence separating the two allows time for students to focus on one same type of rules in one exercise.

The levels of difficulty are summarised in Table 1. Level 1 involves very easy, short exercises using either laws of equivalence: or simple rules of inference. At level 2, students start working on exercises with a mix of laws and rules, and possibly slightly longer exercises. At level 3 they keep working with the same rules but on more difficult exercises. Two rules of inference are left to a later stage because they are much more complex to understand as they involve adding and then removing premises (ie making hypotheses and arriving at a deduction or at a contradiction): the Conditional Proof (CP) and the Indirect Proof (IP). These are left for level 4 onwards. The difference between levels 4 and 5 is the length of the proof.

The progression through these levels is summarised in Table 2. The performance level '3' refers to the completion of an exercise, with or without useless steps or mistakes along the way. That is, if the student manages to find a solution but made some mistakes or useless steps, the performance will be 3. The third column indicates the minimal number of questions they must complete from the current level, with at least the given performance. The last column specifies that the student must have used correctly the specified rules. For example, a student may progress from level 4 to 5 if s/he has completed at least 3 level 4-exercises (with or without mistakes and useless steps) and used correctly the CP and IP rules at least once each.

Table 1
Levels in the curriculum

1	Laws of equivalence OR simple rules of inference, less than 6 lines, less than 4 rules in each exercise
2	Laws of equivalence AND simple rules of inference, less than 7 lines, less than 7 rules in each exercise
3	Laws of equivalence AND simple rules of inference, less than 15 lines and 15 rules in each exercise
4	Any rule, including CP and IP, less than 8 steps and 8 rules in each exercise
5	Any rule, more than 8 rules and steps

Table 2
Level progression criteria

	Min performance	Nb questions	Rules that must have been used correctly
Level 1->2	3	3	
Level 2->3	3	5	
Level 3->4	3	4	
Level 4->5	3	3	CP and IP

In retrospect, the progression we defined was too slow. We could have made the number of questions at each level of 2 (and maintained the same performance). In fact a small proportion of students progressed to the higher levels (10% only). However, although the level is used by the system to retrieve an exercise if the student elects to be suggested one, the impact of this is tempered by the fact that students have the freedom to select or create any exercise, regardless of the difficulty. Note that the criteria values shown in

Table **1** and Table 2 can be changed via the LT-Configurator.

Logic Tutor

The tool was made available and demonstrated to the students during the first lecture on logic, in 2001 (Unix version), 2002 and 2003 (web-based version). In all three years, students had to submit their homework and their assignments from the Logic Tutor. This was not only a way to enforce its use but also had the advantage of significantly reducing marking time and marking errors.

It was interesting to witness that there were a small handful of students complaining that the system was “not functioning correctly” when in fact they were repetitively trying to force the system to accept their mistakes, disregarding the system's feedback. From a computer design point of view, this obviously raised the question that the interface was not appropriate for everyone and could be improved. But more interestingly, from a teaching point of view, it highlighted the fact that students can have deep misconceptions and can be totally oblivious of them, even when they are confronted with evidence to the contrary. The benefit we saw was that these cases were brought up to a tutor or to the lecturer (either because the student would “complain” or because his/her pen rectification would stand out on the print out). So the remedial

discussion would take place with the lecturer or tutor being aware of this ingrained misunderstanding. This also highlights the fact that some students need human feedback and not only computerised ones, and, from our point of view, that the interface could be improved.

Naturally, students were warned that information about their interactions would be stored on the server and would be used for personalization and teaching purposes. Students actually have access to that information, stored in their user model, and can access it at anytime for reflection purposes. No one objected.

LT-Analyser

The LT-Analyser was used in two ways. First, it was consulted during the course of the teaching semester, to find out how the class was going and adjust as much as possible the content of the lectures to the current class. Second, it was queried between two semesters of teaching, to find out information that could help in designing the next course in a better way and to be more proactive about students' common difficulties.

Use during the teaching period. Regularly throughout the period where students used the Logic Tutor, and in particular prior to lectures, I consulted the student database, updated with the most recent students' user models. Through SQL queries and diagrams, I queried the database with the purpose of identifying the most common mistakes and the logic rules causing the most problems giving me a general feeling of how well the class is going. Only class-wide and tutorial group-wide queries were made. I did not make queries to isolate individual students who had problems.

Mistakes analysis: In 2002, out of 2746 mistakes in total the most frequent mistake was the *Premise set incorrect*, in particular with the *Indirect Proof* or *Conditional Proof* rules. These were also two very frequently misused rules (71% and 59% respectively). This is due to the fact that they are the most difficult to grasp because they both require the assumption of an additional premise (for example the negation of the conclusion for *Indirect Proof*, aka proof by contradiction) and then the removal of this premise to reach the conclusion. The highest absolute number of mistakes were with *Modus Ponens* (335) and *Simplification* (273). However they were also the most frequently used rule. In the end they were used incorrectly 30% and 26 % of the time.

Analysis of exercises: not surprisingly, the exercises that produced the most mistakes were the ones involving Indirect and Conditional proofs. However, they were also attempted by a larger proportion of students. Some of these exercises were actually part of homework and assignment.

Student levels: Status on students' progress: more than half the class stayed on level 1 (this included students who only logged in once or twice), then 20% moved to level 2 and 10% reached level 3, 4 and 5. The number of times each student logged in the system ranged from 0 to16.

These findings were mostly useful for the revision lectures. Since lecture time does not allow reviewing everything, I was able to focus on the mistakes made the most frequently and the rules used the most incorrectly. In lectures, I re-explained these concepts, with relevant and concrete examples of mistakes made by the students in the past weeks. This occurred in the

context of exercises in where students had to participate. The students' response (an even greater attention than usual!) was a good indicator of the accuracy of the focus.

Use of previous courses' data to improve subsequent courses.

Further analysis: Subsequent analysis was made after the end of the semester, using other techniques to extract more information from the student data. In particular we used association rule to find out the mistakes that often occur together (Merceron & Yacef 2003). These exposed that the concept itself of formal proofs (especially its "formal" side, ie that each column must contain specific information and the way that information is calculated) caused difficulties.

Redesign teaching: The data from 2002 suggested some changes for 2003. The concept itself of formal proofs, which seems to be more difficult to grasp, was introduced earlier. Also, the fundamental difference between laws of equivalence and rules of inference was much more emphasized. In previous years, students learned about all the laws and rules before attacking the concept of formal proofs. In 2003, the concept of formal proofs was introduced just after the laws of equivalence, a second time after the simple rules of inference, a third time after complex rules of inference, and again a fourth time for proving tautologies.

Exploit mistakes from previous years: Students were given counter-examples based on common mistakes of previous years, not only in lectures but also in tutorial exercises, to engage them actively in finding the mistakes. They were given proof fragments with invalid steps (students were aware that they are invalid) and the aim of the exercise consisted in finding the mistakes and explaining why. They were told that the samples came from past years' data.

EVALUATION

We are presenting here the quantitative evaluation results of the impact of the Logic-ITA on student performance. Then we will present some qualitative results, in terms of student point of view and teacher point of view.

Evaluation of the learning impact

Assessment used

We used, within the normal assessment of the course, the assessment items that were only concerned with logic formal proofs. In all four years, we gave similar homework as well as a similar exam question on formal proofs. The homework consisted of logic proofs, given the premises and the conclusion. The exam question was a logic proof to complete with some parts of the proof provided, and the student had to fill in the missing parts. As for the assignment (from 2002), students were given an argument in plain English and had to prove or disprove its validity. In addition to the actual proof, students therefore had to translate an English argument into a formal logic one.

In 2000, all assessments were paper-pen. From 2001, the homework and assignment had to be done through the Logic Tutor whilst of course the exam question remained paper pen. Hence

the exam question is our most important and unbiased indicator, as homework answers were "filtered" by the Logic Tutor, whereas the exam question was not.

Results

We compared the results over these assessments for the 2003 and 2002 classes (who used the web-based Logic Tutor), the 2001 class (who used the Unix Logic Tutor), and the 2000 class (who did not use the tool at all). Table 3 shows various measures of dispersion for these tests, over the three years. The first line shows the mean and, in brackets, the standard deviation. The second line shows the number of scores and the median. Marks are brought to the same scale throughout the 3 years: homework scores are out of 3, exam question out of 7 and the assignment is out of 5. Students who did not sit the exam or did not hand in their homework are not counted.

Table 3
Measures of dispersion

	2000	2001	2002	2003
Homework (3)	1.9 (0.9) N=242, Med=2	2.3 (0.9) N= 244, Med=3	2.9 (0.4) N=184, Med=3	2.4 (0.7) N=106, Med=3
Assignment (5)	N/A	N/A	4.8 (0.5)	4.9 (0.3)
Exam question (7)	3.3 (1.6) N=431, Med=3	4.2 (2.7) N=390, Med=5	4.7 (2.8) N=245, Med=6	5.2 (2.4) N=132, Med=7

We can see that the means and median scores increase each year. The average marks for homework and assignment from 2001, not surprisingly, are very high, strongly due to the fact that the Logic Tutor takes care of the mistakes before the submission, so that makes the results not very informative. However there has been a clear increase in the exam question on logic.

Table 4 shows the results of ANOVA on these exam question scores for each year where the Logic-ITA was used versus the control year 2000. The last column shows the effect size using the Glass's delta, as it is more commonly used in educational studies. It is computed as the difference between the mean scores of experimental and control groups, divided by the standard deviation of the control group. Effect size provides a common scale that standardizes the various measurements used in different studies. Whilst Bloom reports studies showing that human tutoring can yield an effect size of 2 sigmas (i.e. standard deviations of the control group), Kulik's meta-analytic study (Kulik 1994) of computer assisted instruction reports that, over a wide range of instructional areas and student levels, a gain of approximately .35-sigma is achieved. Koedinger, Anderson and colleagues have also shown in various evaluation studies that their cognitive tutors reached a 1 sigma effect (Anderson, Corbett, Koedinger & Pelletier 1995; Koedinger & Anderson 1997) if not more (Corbett 2001).

Results in these two tables show that there was a steady increase each year in the exam results and hence in the effect size and ANOVA showed a statistic difference between the results ($p<0.001$). The 2001 students scored higher than 2000 students, with an effect size of 0.6 sigma. In 2002 students scored higher again, reaching an effect size of 0.9 sigma over the control group and in 2003 the effect size was 1.2.

Table 4
Analysis of variance and effect size

Comparison of exam question results	F value ($p<0.001$)	Effect size
2000-2001	F=4.9	0.6
2000-2002	F=69.9	0.9
2000-2003	F=114.2	1.2

We also looked at the comparison of results in another exam question, which remained similar throughout the years (except for an additional sub-question in 2003), and which is related to a completely different part of the course. It showed that the results were similar from year to year: 10.5 (in 2000), 9.5 (in 2001), 10.3 (in 2002) and 8.5 (in 2003) out of 15. The change of lecturer in 2001 and the restructure of the school curriculum did not seem to have a significant impact on this part of the course.

We were particularly interested in the regular increase in effect size. There seems to be a snowball effect. In each year, the teaching was improved in light of the findings in the student data and this seems to reflect on student performance at the exam.

Looking at individual comparison from each year with the control year, an average of 80% of students in 2003, 65% of students in 2002 and 50% in 2001 exceeded the levels of achievement attained by only 10% of students in 2000. This indicates to us that students were better prepared. If nothing else, the Logic-ITA gave students the opportunity to practice with feedback more often, which means they were more familiar with the process of solving formal proofs prior to sitting the exam. This is also indicated by the results shown in Table 5 where results are shown according to the number of exercises they attempted. The right column indicates the mean, the standard deviation in parentheses and the number of scores.

Table 5
Breakdown of exam question results per student activity
on the Logic Tutor (year 2002)

Activity on the Logic Tutor	Results
More than 10 exercises	5.4 (2.5), N=36
6 to 9 exercises	5.1 (2.7), N=48
3 to 5 exercises	5.0 (2.8), N=56
0 to 2 exercises	4.2 (2.9), N=-105

As we can see, students who used the tool the most received higher marks. However, we cannot ascertain whether this slight increase is due to the use of the Logic-ITA or simply to the likeliness that the better students, who would do well anyway, are also the ones that use the Logic Tutor because they are generally more motivated. The number of exercises attempted by the students ranged between 0 and 32 and averaged 8. There is a large proportion of students who only did 0, 1 or 2 exercises. A contributing factor would be the fact that many students chose to do their homework in pairs (hence only one person is logged in).

The aim of the experiment was first to implement a tutoring system which feeds back information to the lecturer and then to investigate whether this tutoring system, combined with the greater information that the lecturer would gain about the current and previous classes, would

improve learning and teaching. These results are consistent with this idea. Over the years of use of the Logic-ITA, the overall student performance has increased each year.

There are other possible explanations for this increase. For instance the change in student population (especially from 2001 to 2002 where students had a slightly larger choice of subjects) as well as the change of lecturer from the original control group (2000) can also be seen as contributing factors for this improvement. However, as mentioned earlier in this section, the fact that the results in other parts of the course remained similar does not support this alternative explanation.

It is important to note that the evaluation reflects on the *whole* usage of the Logic-ITA, without highlighting which proportion is due to the Logic Tutor itself, and which proportion is due to the monitoring process, leading to the teacher's greater awareness and understanding of the class's difficulties. This tool invites the teacher to spend more time thinking about teaching and learning issues, during and after teaching semesters. Hence the teaching is bound to evolve throughout the years, hopefully leading to better learning for the students. We cannot identify how much of this is due to the Logic-ITA, but we can say that the Logic-ITA supports this reflection process.

Qualitative evaluation

The Logic Tutor

We conducted a student evaluation survey on a voluntary basis with students, teachers and tutors. In July 2001, before the tool was used, 9 people participated in a survey. The population consisted of 1 teacher (the former teacher for the course), 4 tutors and 4 students. These tutors and students took the course in earlier years. The survey related to various aspects of the Logic Tutor (interface, help, usability, usefulness) and allowed free comments. In November 2001 and in November 2002, the population surveyed consisted exclusively of students who attended the course and used the tool. The survey was done in the context of general course surveys that students are encouraged to fill in at the end of the teaching semester (so although the survey was of voluntary nature, it was not specific to the Logic Tutor). Unfortunately not many students usually take the time to fill out the course surveys (here only 5-10% of students took the time to complete it). The survey in 2001 had similar questions to the pre-survey whereas in 2002 students were only asked to give their opinion on the usefulness of the tool. Those who wanted to add comments could do so in a free text section. Unfortunately the course survey for 2003 was not conducted.

In 2001 the feedback from the students was positive apart from the fact that they would have liked the freedom of using the tool from various locations. In 2002, this downside had disappeared and students thought the tool helped them understand logic proofs better.

As we can see in Table 6, students make clear that they like the opportunity to practice with a system as long as it is not their primary source of learning. The last comment on the low satisfaction in first time use links with a result we found using symbolic data analysis. Students seem to be slow at using the system, but once they do at least 2 exercises, they are likely to do more. This highlights an initial barrier of use.

Table 6
Extracts from students' comments

Is it useful in learning propositional logic?

- "Excellent to learn from, since the program offers helpful suggestions about what action to take."
- "Computers no substitute for human teachers!"
- "Yes. It forces the students to construct correct proofs."

Would you like to see this type of tool used more in courses?

- "Sure! Please, it would definitely help the helpless people like me."
- "Yes. Self learning and practice is good."
- "Yes – as an extra resource the students can use…but not as the primary source of their learning."

What is good?

- Statistics and mistake history - help identify problems.
- Question database.
- Can do as many questions as we want and can get important feedback on them.
- Feedback - is context sensitive and hints are very appropriate.
- Forces students to construct correct proofs without missing any steps.

What is bad? What needs to be improved?

- Usability - Needs to be improved.
- Help files - Need access to help while doing question. Also good if one could print the help pages out. Also good if there was some help explaining what each of the input fields were (for first time users).
- Level of satisfaction from using the program is low to begin with.

The LT-Analyser

Here I can only report on my personal experience as the teacher for the course. As explained in the previous section, I used the LT-Analyser twice a week during the use of the Logic Tutor as well as analysing student data between semesters. I had saved a number of queries, such as the breakdown of mistakes per type of mistake and per rule and used the results to adapt the lecture contents and warn tutors of problem areas to look for. The students' response when I focused on the major problem areas indicated to me that this was extremely useful. I also found it useful to be aware of the real common problems rather than guessing what they could be. It also gave me concrete guidelines to amend the next year course and to design more appropriate material. Much more detailed analysis was carried out outside teaching time and revealed very interesting patterns in learning (Merceron & Yacef 2003; Merceron & Yacef 2004).

One drawback of the system was that analysing the data can take time (unless there are just a repetition of saved queries). This is only a technical problem that can quite easily be solved with a more powerful and user-friendly analysis software.

Another drawback was also that the experiment was conducted on a small topic, which only covered 2 weeks of lectures. This means that we could not take full advantage of the teacher feedback functionality. I could give feedback to the students in the second week, and then during the revision lecture at the end of the semester. One can imagine that, had the topic been larger and covering a longer period, the benefits for the teacher would have been greater.

CONCLUSION

The Logic-ITA is an Intelligent Teaching Assistant system in the domain of logic. It aims at alleviating some of the problems caused by large classes or distance learning, acting as a complementary intermediary between teacher and students: on one hand it provides students with an environment to practice formal proofs with immediate feedback and on another hand it allows teachers to monitor how the class is progressing. It is complementary in the sense that it does not aim at replacing any of the existing interactions between teachers and students but complement them. In this sense, it bypasses some of the criticisms that are made about ITSs and ILEs such as their strong individual tutoring focus (Andriessen & Sandberg 1999; Cumming & McDougall 2000; Laurillard 2002) or the narrowness of their teaching strategies and tactics (du Boulay & Luckin. 2001). The Logic-ITA is a tool that empowers teachers to do their work better. It only automates a few teaching tactics (eg provision of immediate feedback) where the teacher is not able to provide enough of them, due to time and resource constraints, and records for the teacher the class's activity, mistakes and progress. In the whole teaching and learning process, the roles of the teacher and students are preserved. The Logic-ITA's role is to service them, without the goal to take any control of the process.

The interesting part of this study is the combination of an ordinary intelligent tutor with components for the teacher to assist classroom teaching. We have presented a summary of how the tool works and conducted an evaluation comprising of four groups: the 2002 and 2003 classes, who used the web-based Logic Tutor, the 2001 class who used the former Unix version and the 2000 class, who did not used the tool at all. Results were very encouraging and showed an effect size on students' marks increasing each year from 0.6 to 1.2 in the exam question on logic.

There are still some functionalities of the Logic-ITA that need to be tested and evaluated. In particular, the LT-Analyser can be used to identify individual students who are struggling in order to intervene. We have only performed group monitoring. We plan to carry out individual monitoring in the next course session. We are also working on an additional layer of intelligent assistance to mine and visualize the student data. Our aim is that this component will be able to detect specific or abnormal patterns and alert the teacher about them.

ACKNOWLEDGEMENTS

The author thanks Agathe Merceron, David Abraham, Elisabeth Crawford and Leanna Lesta for their various contributions to the project. This project was funded by a University of Sydney Sesqui grant.

REFERENCES

Abraham, D., Crawford, L., Lesta, L., Merceron, A., & Yacef, K. (2001). The Logic Tutor: a Multimedia Presentation. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 3(2).

Abraham, D., & Yacef, K. (2002). Adaptation in the Web-based Logic-ITA. In P. de Bra (Ed.) *2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH2002)* (pp. 456-461). Malaga, Spain, Springer-Verlag.

Anderson, J. R. (1983). *The architecture of cognition*. London, Harvard University Press.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *The Journal of the Learning Sciences,* 4(2), 167-207.

Andriessen, J., & Sandberg, J. (1999). Where is Education Heading and How About AI? *International Journal of Artificial Intelligence in Education,* 10, 130-150.

Boy, G. A. (1991). *Intelligent Assistant Systems.* London: Academic Press.

Corbett, A. T. (2001). Cognitive computer tutors: Solving the two-sigma problem. In M. Bauer, P. J. Gmytrasiewicz & J. Vassileva (Eds.) *User Modeling (UM2001)* (pp. 137-147*)*. Sonthofen, Germany. Berlin: Springer.

Croy, M. (1999). Graphic Interface Design and Deductive Proof Construction. *Journal of Computers in Mathematics and Science Teaching*, 18(4), 371-386.

Croy, M. J. (1989). CAI and Empirical Explorations of Deductive Proof Construction. *The Computers and Philosophy Newsletter*, 4, 111-127.

Cumming, G., & McDougall, A. (2000). Mainstreaming AIED into Education? *International Journal of Artificial Intelligence in Education,* 11, 197-207.

du Boulay, B., & Luckin., R. (2001). Modelling human teaching tactics and strategies for tutoring systems. *International Journal of Artificial Intelligence in Education* 12(3), 235-256.

Fitts, P. M., & Posner, M. I. (1967). *Human Performance*. Belmont, CA: Brooks Cole.

Habeshaw, S., Gibbs, G., & Habshaw, T. (1992). *53 Problems with Large Classes. Making the Best of a Bad Job*. Bristol, Technical & Educational Services.

Hara, N., & Kling, R. (1999). Students' Frustrations with a Web-Based Distance Education Course. *First Monday,* 4(12).

Jean, S. (2000). *Pépite: un système d'assistance au diagnostic de compétences*, PhD, University of Le Mans, Le Mans.

Kay, J. (2000). Accretion representation for scrutable student modelling. In G. Gauthier, C. Frasson & K. VanLehn (Eds.) *Intelligent Tutoring Systems (ITS'2000)* (pp. 514-523). Berlin: Springer-Verlag.

Kinshuk, Tretiakov, A., Hong, H., & Patel, A. (2001). Human Teacher in Intelligent Tutoring System: A Forgotten Entity! In T. Okamoto et al (Eds.) *Advanced Learning Technology: Issues, Achievements and Challenges*. Los Alamitos, CA: IEEE Computer Society.

Koedinger, K. R., & Anderson, J. R. (1997). Intelligent Tutoring Goes to School in the Big City. *International Journal of Artificial Intelligence in Education*, 8, 30-43.

Kulik, J. A. (1994). Meta-analytic studies of findings on computer-based instruction. In E. Baker & H. O'Neil (Eds.) *Technology assessment in education and training*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Laurillard, D. (2002). *Rethinking University Teaching: A Framework for the Effective Use of Educational Technology*. London: Routledge Flamer.

Leroux, P., Vivet, M., & Brezillon, P. (1996). Cooperation between a Pedagogical Assistant, a Group of Learners and a Teacher. *European Conference on AI in Education* (pp. 379-385*)*. Lisbon, Portugal.

Lesta, L., & Yacef, K. (2002). An Intelligent Teaching-Assistant System for Logic. In S. Cerri & F. Paraguo (Eds.) *International Conference on Intelligent Tutoring Systems (ITS'02*) (pp. 421-431*)*. Biarritz, France. Berlin: Springer-Verlag.

Mark, M. A., & Greer, J. E. (1995). The VCR tutor: Effective Instruction for device operation. *The Journal of the Learning Sciences,* 4(2), 209-246.

Merceron, A., & Yacef, K. (2003). A Web-based Tutoring Tool with Mining Facilities to Improve Learning and Teaching. In F. Verdejo & U. Hoppe (Eds.) *11th International Conference on Artificial Intelligence in Education* (pp. 201-208*).*Amsterdam: IOS Press.

Merceron, A., & Yacef, K. (2004). Clustering students to help evaluate learning. In J.-P. Courtat, C. Davarakis & T. Villemur (Eds.) *Proceedings of TeL'04 - Technology Enhanced Learning, 18th IFIP World Computer Congress* (pp. 31-42*).* Toulouse, France. Kluwer Press,

Pittinsky, M. S. (2003). *The Wired Tower: Perspectives on the Impact of the Internet on Higher Education.* Englewood Cliffs: Prentice Hall.

Rasmussen, J. (1983). Skills, rules and knowledge : signals, signs and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics,* 3, 257-266.

Scheines, R., & Sieg, W. (1994). Computer environments for proof construction. *Interactive Learning Environments*, 4(2), 159-169.

Sweller, J. (1993). Some cognitive processes and their consequences for the organisation and presentation of information. *Australian Journal of Psychology,* 45(1), 1-8.

Van Labeke, N. (1999). *Prise en compte de l'usager enseignant dans la conception des EIAO, Illustration dans Calques 3D*. PhD dissertation, Universite Henri Poincare, Nancy I, Nancy.

Virvou, M., & Moundridou, M. (2001). Adding an Instructor Modelling Component to the Architecture of ITS Authoring Tools. *International Journal of Artificial Intelligence in Education*, 12, 185-211.

Vivet, M. (1992). Uses of ITS: which role for the teacher? In E. Costa (Ed.) *New Directions for Intelligent Tutoring Systems* (F91). Berlin, Heidelberg, New York: Springer-Verlag.

Yacef, K. (1999). *Vers un assistant tutoriel intelligent pour la formation d'opérateurs de systèmes complexes*. PhD dissertation, University of Paris 5, Paris, France.

Yacef, K. (2002). Intelligent Teaching Assistant Systems. In Kinshuk (Ed.) *International Conference on Computers in Education (ICCE'02)* (pp. 136-140). Auckland, New Zealand.