USING COMPUTERISED TESTS AS SUPPORT FOR TUTORIAL-TYPE LEARNING IN LOGIC

Rein PRANK
University of Tartu
Institute of Computer Science
Liivi Str 2, 50409 Tartu, Estonia
e-mail: prank@cs.ut.ee

ABSTRACT

The course "Introduction to Mathematical Logic" (32 h lectures + 32 h exercises + 56 h independent work) is compulsory at the Faculty of Mathematics and Informatics of the University of Tartu. In 1987-1991, we designed four programs for the exercises: Truth-Table Checker, Formula Manipulation Assistant, Proof Editor and Turing Interpreter. As result, we transferred 70% of the exercises to the computer class. The blackboard-based practical training was preserved for predicate logic. In the subsequent years we added one more program for truth-values of predicate formulas. The main benefit of the first round of computerization was the acquisition of real skills in two areas: Turing machine programming and proof construction. At the same time, rejection of blackboard exercises reduced the possibility of assigning to students small but important questions concerning new topics.

In 1998, we decided to use a test administration system APSTest to introduce tutorial-type support for lectures. After each 1-2 lectures, a test comprising 10-15 questions is available in the faculty network. The test has randomly selected questions and can be solved a number of times; before the beginning of the next lecture, however, the students shall have achieved at least a 75% success rate. The weekly duty to keep oneself up to date with theoretical material has reduced the dropout rate and improved the average grade results.

The paper describes:

- Test system APSTest and our test structure,
- Intent of questions for different types of lecture topics,
- The use of different types of questions,
- Eligibility of test-type questions for the different parts of the course,
- Some conclusions about the students' learning strategies drawn on the basis of the data saved in the database,
- Topics and questions proving to be more difficult,
- Changes made over time in test management.

KEYWORDS: Mathematical Logic, Computer Assisted Assessment

1. Introduction

For any course of lectures to be efficient, it is necessary that the students familiarize themselves with the material already covered by doing independent work. Written exercises used in teaching mathematical subjects do not satisfy the need fully, for they usually deal with technical problems requiring a longer solution. The computerization of exercises may even aggravate the problem, for the problems and questions that the existing software is incapable of addressing may be disregarded altogether. An introductory course of mathematical logic has some qualities that increase the need for tutorial-type work. The course introduces a large number of new but relatively simple concepts. Concepts, formulas, canonical forms, rules, etc. are created by interrelated groups. The elements of a group develop fairly similar relationships with one other as well as with other objects. Consequently, only a small part of them are analysed in lectures and textbooks while the remaining part is left to the students themselves to prove by analogy, or sometimes even to discover and invent on their own.

This article describes the use of computerised tests to support the weekly independent work required for the learning of the theoretical material contained in the introductory course of Mathematical Logic taught at the Faculty of Mathematics and Informatics of the University of Tartu. During a semester, the students independently take 10-12 tests in the computer class. The tests are generated from a databank currently containing approximately 500 questions. The system has been used for three years, which allows us to evaluate both the questions and their effect on the learning of the discipline.

Part 2 of the article gives an overview of the essence of the course as well as the problem-solving software created in previous years. Part 3 describes some features of the test administration package APSTest used in the work, and our organization of tests. Part 4 describes the questions used in the tests and examines the topics of the course they cover. Part 5 investigates, on the basis of the data stored in the database, students' working strategies in doing tests, including inappropriate behavioural patterns. Part 6 evaluates the current situation in the implementation of the system. In comparison with (Croft, Danson, Dawson & Ward , 2001), our experiment is more directed to knowledge and less to skills.

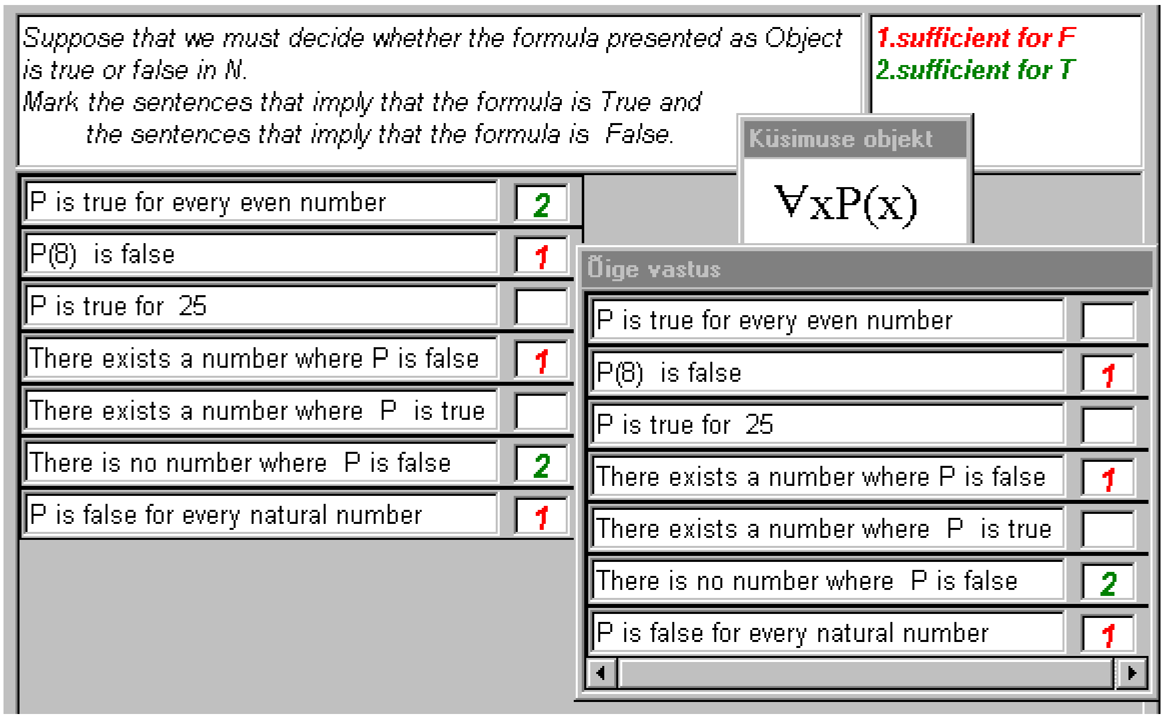
2. The course and computerised exercises

The course Introduction to Mathematical Logic has been on the curriculum of the Faculty of Mathematics and Informatics of the University of Tartu for some time already. The course is compulsory, and most of the students take it in the spring term of their second year. 70-90 students usually attend the course. The discipline has been planned to consist of 32 hours of lectures, 32 hours of workshops and 56 hours of independent work. The lecture themes of the course are presented in Table 1.

In 1987-1991, we designed four computer programs for doing exercises: Truth-Table Checker, Formula Manipulation Assistant, Proof Editor and Turing Interpreter (Prank 1991). The main purpose of the work was to create problem-solving environments for two difficult domains – Formal Proofs and Turing Machines. In addition, the two first mentioned programs were designed for computerising the main problem types contained in the first chapter of the course. As a result, we transferred 70% of the exercises to the computer class. The blackboard-based practical training was restricted to predicate logic.

Introduction to Mathematical Logic **Lectures (16 × 2h)**
I. Propositional Logic (Model theory) 1. Introduction. Sentences, truth-values, propositional connections, formulas, truth-tables. 2. Tautologies, satisfiability, logical equivalence. Main equivalences. 3. Expressibility by $\{\&, \neg\}$, $\{\vee, \neg\}$, $\{\supset, \neg\}$. Normal forms and their properties.
II. Predicate Logic (Model Theory) 4. Predicates, quantifiers. Validity of formulas for finite models, N, Z, R. 5. Signature, first-order language, interpretation, expressibility. 6. Tautologies, logical equivalence. Main equivalences. Prenex form. 7. Proofs of main equivalences.
III. Axiomatic Theories. 8. Introduction. Axioms and rules of propositional calculus. Examples of the proofs. 9. Consistence. Implications for proof-building. 10. Completeness of propositional calculus. 11. Predicate calculus. Examples of the proofs. Consistence. Completeness (without proof). 12. First-order axiomatic theories. Group theory. Formal arithmetic.
IV. Algorithm Theory 13. Introduction: Concrete algorithms and algorithm theory. Turing Machine. Computing numerical functions on TM. 14. Operations with TM (composition, branching). 15. Enumeration of TM. Halting problem. 16. Overview of decidability problems in mathematics and computer science.

Table 1. Themes of Lectures


Suppose that we must decide whether the formula presented as Object is true or false in N.
Mark the sentences that imply that the formula is True and the sentences that imply that the formula is False.

1.sufficient for F
2.sufficient for T

Küsimuse objekt

$\forall x P(x)$

P is true for every even number	2
P(8) is false	1
P is true for 25	
There exists a number where P is false	1
There exists a number where P is true	
There is no number where P is false	2
P is false for every natural number	1

Õige vastus

P is true for every even number	
P(8) is false	1
P is true for 25	
There exists a number where P is false	1
There exists a number where P is true	
There is no number where P is false	2
P is false for every natural number	1

Figure 1. A class-assigning question with student's and correct answer.

In the subsequent years, we have made small improvements to our programs, and renewed the user interface. In addition, we added a new program for exercises on the interpretations of predicate formulas. The computerisation of exercises indeed contributed to the learning of the central concepts of the last two chapters, rendering the concepts less abstract through their practical use. At the same time, however, the students' interest in computer exercises tended to eclipse the deeper meaning of the discipline, for the learning of which the exercises were created in the first place. There arose a need to find a better balance between the lecture material and the exercises in students' work.

3. The test system APSTest and our test structure

In 1998, the test administration package APSTest was created as one of the software projects within the state-funded programme aimed at the computerization of Estonian schools. A characteristic feature of APSTest is the availability of a large number of question types:

1) Yes/no questions,
2) Multiple-choice questions (in a list or between the words of a text),
3) Marking questions (in a list or between the words of a text),
4) Matching questions,
5) Class assigning questions (Figure 1),
6) Grouping questions,
7) Sequencing questions,
8) Filling the blanks (with free input or using multiple-choice),
9) Short-answer questions,
10) Numerical questions.

The program enables to vary many characteristics of tests and to compile tests for different purposes. APSTest saves the following data for each try: the time (beginning and duration), the points scored and the success rate, the number of correct, nearly correct and incorrect answers, the questions asked, the time spent on each question and the answers given. Using queries, the teacher can then build tables concerning the data of interest to him. It allows the teacher to relatively simply draw conclusions about both the work done by a particular student and the level of skills mastered in different domains of the discipline, as well as to pass judgements on the level of questions and the general characteristics of the tests. APSTest runs under Windows. The data can be stored in different SQL-based database systems.

Following the launch of the APSTest package, we decided to test its applicability to providing tutorial-type support to lectures. Within the space of a few years, the following test structure has developed. After each 1-2 lectures, a test of 10-15 questions is put out. It can be solved in the computer classes of the Faculty. A test contains randomly selected questions, and it can be solved several times; before the beginning of the next lecture, however, the students shall have achieved at least a 75% success rate. The time for doing a weekly test is not limited. For finding answers, the students are advised to examine the lecture notes and the respective literature. Cooperation between students is also allowed. Thus, two or three students doing tests on adjacent computers while consulting with each other and studying their lecture notes is a regular sight in computer classes. The average grade for the tests accounts for 10% of the total grade for the course. At the end of a semester, a summary test is conducted on the entire material (approximately 30 questions), which also accounts for 10% of the total grade.

4. The questions

This part of the article describes the questions: for what purpose and for what topics of the course they were composed and in what form they were presented. In the two first chapters of the course, each lecture introduces no less than whole series of new concepts. Thereafter, a lecture usually deals with just a few characteristic cases for each issue; the rest of the material needs to be learned by the students themselves using analogy. The students need to learn and memorize tens of equivalences binding different logical operations and quantifiers as well as a number of derivation rules and proof strategies. The best method of mastering this knowledge is exercises where concepts, formulas and other things need to be compared with each another and applied. First steps in this work can be presented as test questions applicable to achieving different educational aims.

1. The definitions of new concepts formalize certain ideas about sentences, truth-values, proofs, algorithms, etc. Test questions can be used to make the students think about what we have actually postulated and what choices we have made. Let us give an example of a question about the concept of sentence:

Many hypotheses of unknown validity have been formulated in mathematics.
What is the mathematicians' attitude towards such sentences?

1) They do not have any truth- value; therefore they are not sentences.

2) They actually have a logical value, and the fact that we do not know it is not important.

3) They can be considered sentences in terms of Propositional Calculus if their logical value is established.

2. Questions concerning the exact wording of definitions, rules, etc. can be asked using multiple-choice blank filling. This is particularly appropriate when several similar concepts, equivalences, etc. are being studied.

3. After a concept (formula, rule) has been introduced, the students can use test questions for training its execution in direct (1-2-step) applications.

Which of the following figures can be the exponent of 3 in the Gödel number of command of the Turing Machine? 0/1/2/3/4/5

Figure 1 shows a quite difficult class-assigning question concerning the universal quantifier.

4. Some test questions are also applicable to comparing similar and interrelated concepts (formulas, equivalences) and finding relationships between them. In addition, they facilitate the distinguishing of valid principles from their invalid analogues:

Mark the pairs of equivalent formulas:
$\forall x(A(x)\&B(x)) \equiv \forall xA(x)\&\forall xB(x)$? $\forall x(A(x)\vee B(x)) \equiv \forall xA(x)\vee\forall xB(x)$?
$\forall x(A(x)\supset B(x)) \equiv \forall xA(x)\supset\forall xB(x)$? $\forall x(A(x)\sim B(x)) \equiv \forall xA(x)\sim\forall xB(x)$?

5. Some questions are also applicable to giving concrete examples of the relationships between mathematical logic and other branches of mathematics:

Mark the operations on the set of rational numbers that are applicable to interpreting some binary functional symbol f(x,y):
$x+y$, $x-y$, $x\cdot y$, $x: y$, x^y

6. Students with only a superficial acquaintance with a certain problem contained in the course (such as manipulation to normal form) know in general what operations need to be performed for solving the problem. Sequencing questions are applicable to inquiring them about the sequence of steps in an algorithm as well.

7. Matching questions are applicable to building formulas from the "prescribed material":

Express the formula $\forall xR(x,y)\supset P(y)$ *in prenex form, matching the necessary strings and symbols with numbers 1, 2, 3, ... and leaving the rest unpaired:*

$\forall x$, $\exists x$, $\forall y$, $\exists y$, $R(x,y)$, $P(y)$, (,), $\supset$
1, 2, 3, 4, 5, 6, 7

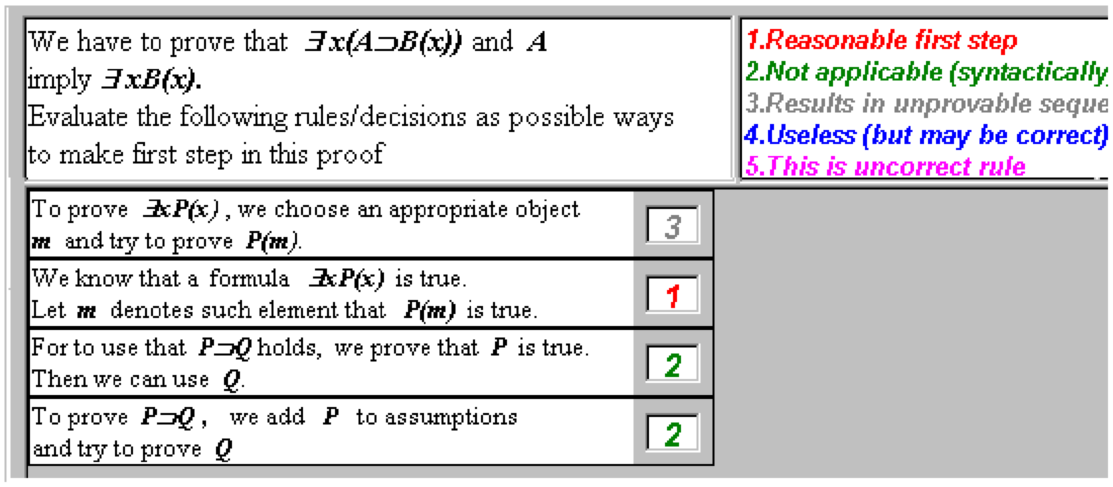
8. Watching the students solve proof problems led us to the idea of supporting proof building by test questions on the "local" problem. While building the proof tree from root to leaves, it is possible to apply some rules to the sequence under construction in such a way as to generate a sequence above the line that is not valid and therefore cannot be proved. Insofar as predicate logic is concerned, the program is unable to diagnose the errors, and if the student does not notice his error himself, he will try to solve an insoluble problem from that point on. To direct attention to the possible effects of the steps, we gathered material concerning the mistakes actually made in the solutions, and, after examining other possible proof tasks in predicate logic, added a large number of questions concerning analogous situations. One example is given in Figure 2.

Let us now examine the use of weekly tests on different parts of the course. The first two chapters of the course deal with the introduction of the languages of Propositional Logic and Predicate Logic, their interpretations, main equivalences, inferences, different canonical forms, etc., their simple applications and their relationships to different domains of mathematics. Accordingly, test-type questions are very suitable for achieving many educational aims on these themes. Therefore, abundant use of questions is made in teaching the material of the first six lectures. The bulk of the more voluminous exercises (problems based on truth-tables, formula manipulation and the logical value of predicate formulas in concrete interpretations) are solved in computer class during workshops or as independent work. Blackboard-based workshops are used for expressing propositions through formulas and doing exercises on inference and equivalence proof/disproof.

In the chapter on axiomatic theories, the number of questions to be tackled is much smaller. The rules of Grentzen-type Propositional Calculus and Predicate Calculus are introduced and argued. The bulk of the lecture time is spent on the proof of the properties of the systems. The building of formal proofs is virtually the only type of tasks solved in classrooms. We have special software for that. The first test is conducted on the material taught at the first lecture of the chapter and it covers the general concepts and rule properties. At the end of the chapter, two tests are solved, where problems of step selection described under Item 8 are supplemented with those dealing with the properties of quantifier rules and concrete first-order theories.

In a similar manner, the chapter on algorithm theory has been built around one central concept. In workshops, Turing machines are constructed for calculating various numerical functions using unary and binary codes. Two tests are solved, of which the first one is built primarily on the material presented in the introduction while the second deals with the specifics of the enumeration of Turing machines.

As concerns the types of the questions used, the current database of approximately 500 questions contains marking questions (42%) and class assigning questions (41%), along with multiple-choice and matching questions (both 5%), numerical and short-answer questions (both more than 2%), and all the remaining types (each less than 1%). Such a distribution of types is apparently attributable to the authors' intention to support theoretical learning and provide questions on comparison, evaluation and classification promoting the making of generalizations rather than just ask for the facts. In the opinion of the author of this paper, such a distribution also testifies to the functionality of the implementation of class assignment questions as a separate type in the test system.

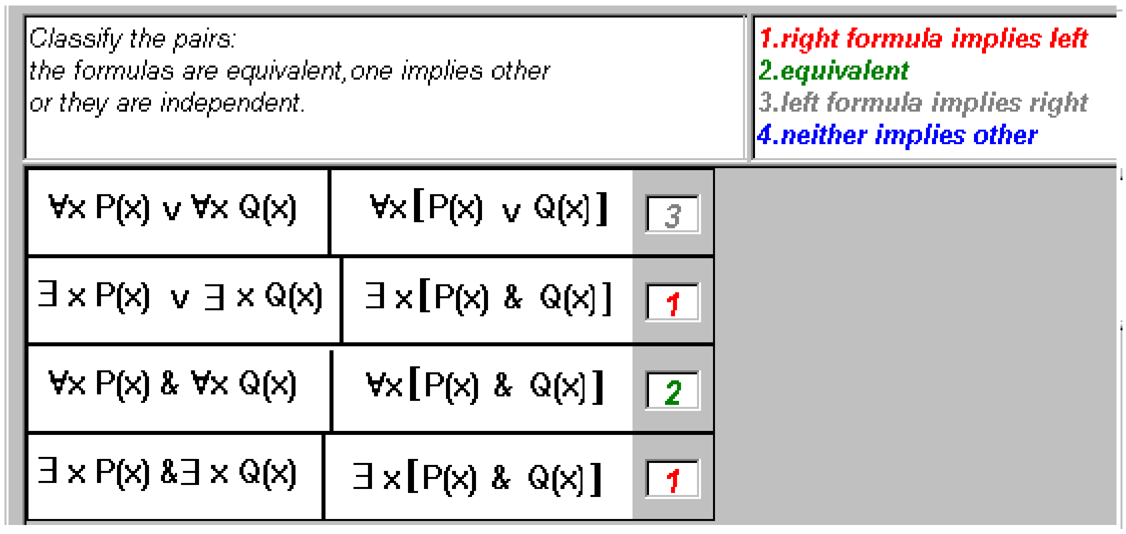


We have to prove that $\exists x(A \supset B(x))$ and A imply $\exists x B(x)$.
Evaluate the following rules/decisions as possible ways to make first step in this proof

1.Reasonable first step
2.Not applicable (syntactically
3.Results in unprovable seque
4.Useless (but may be correct)
5.This is uncorrect rule

To prove $\exists x P(x)$, we choose an appropriate object m and try to prove $P(m)$.	3
We know that a formula $\exists x P(x)$ is true. Let m denotes such element that $P(m)$ is true.	1
For to use that $P \supset Q$ holds, we prove that P is true. Then we can use Q.	2
To prove $P \supset Q$, we add P to assumptions and try to prove Q	2

Figure 2. A question on possible steps in the proof



Classify the pairs:
the formulas are equivalent, one implies other
or they are independent.

1.right formula implies left
2.equivalent
3.left formula implies right
4.neither implies other

$\forall x P(x) \vee \forall x Q(x)$	$\forall x[P(x) \vee Q(x)]$	3
$\exists x P(x) \vee \exists x Q(x)$	$\exists x[P(x) \& Q(x)]$	1
$\forall x P(x) \& \forall x Q(x)$	$\forall x[P(x) \& Q(x)]$	2
$\exists x P(x) \& \exists x Q(x)$	$\exists x[P(x) \& Q(x)]$	1

Figure 3. Questions on the comparison of formulas proved to be difficult

5. Students' working strategies and results. Changes in the organization of tests

First of all, we must speak about a few problems that arose upon the launch of the test system. Technically, these concern exactly the free use of the test system where tests can be solved for an unlimited number of times.

The author composed the first weekly tests of exercises in such a manner that after giving an incorrect answer the student was able to press a button and see the correct answer. However, the students started to look up and write down the correct answers before solving the tests (at that time, the number of parallel questions was still fairly small). Thus, we had to disable the correct answer feature, and the students then need to work on their lecture notes in order to understand their errors.

Next, the students discovered that it is possible to run several copies of APSTest on the same computer at the same time. Running several "trial copies" alongside the "main copy", they were able to try several variants for answering a question, until the program acknowledged one of them as true. The programmers then had to improve the programme to deny them this opportunity.

Based on the analysis of the database, we have discovered different strategies used by students for taking tests allowing an unlimited number of tries.

1. The most noticeable were the students who set the goal of achieving a 100% or nearly a 100% success rate. Stronger students usually needed 1-3 tries for that. They carefully considered each question of each try, spending an average of 0.5-2 minutes per question. Quick reply was only given to relatively simple questions as well as those whose answer was known from a previous try. Occasional bulkier questions took them 3-4 minutes to complete. Some students solved a test 3-4 times even after they had already achieved the maximum score. Their intention was to obtain a better knowledge of the entire material by answering the different variants of the questions. Such an approach, which requires approximately one hour per week, might be considered optimal. For instance, at the spring semester of 2001, 31 students out of 89 had scored a 95% or a better success rate in at least 8 tests out of 10. More than one half of these students have a behavioural pattern similar to that described under this item. At all semesters, the number of students scoring a maximum result is smaller at the beginning and increases with subsequent tests; however, this is mainly due to the students taking more tries.

2. Some students looked through a test once or twice without answering any questions before actually doing it. By doing this, they found out which themes of the lecture were represented in the questions and which were not. After that, they read the lecture notes and then scored a try.

3. A small number of students (less than 10) only tried to solve each test once and made no attempts to score the maximum points. They only took another try if they did not succeed in hitting the required 75% mark. Of that number, the students who were stronger often scored a result that was quite close to the maximum.

4. For students who were weaker than average, it took 5-8 tries to achieve the desired result. Very few of them took a break to examine the lecture notes after an unsuccessful try. Usually, they started a new attempt immediately after they finished the previous one. Due to the large number of tries, they had already memorized some questions by the time they reached the last tries. The tables in the database show that they worked through the last tries mechanically, with the time spent on answering being less than half a minute per questions; in multiple-choice questions it was often just a couple of seconds.

5. There were also students who regularly took 15-20 tries per test. It took them at least three hours. Apparently, they tried to do the test without scrutinizing the theory, giving an incorrect answer to even rather simple questions. Often, such students broke off the test after a weak score from the first questions and started from the beginning again. After a while, they had memorized the correct answers to all the variants of the first questions yet a subsequent set of questions led them to new break offs. As a result, they more or less memorized the answers to the first themes, until it sufficed for achieving the 75% success rate. The last themes of the material, however, were practiced less than the previous ones. In most cases, these students displayed no change of strategy over the course of the semester.

6. Analysis of the current situation

The experience gleaned from the three academic years has shown that the tests provide support for students in learning the course. The students have rated the use of computer tests as positive in both informal conversations and formal questionnaires. The need to do tests requires periodical work on lecture notes, which, in turn, improves understanding at subsequent lectures. There has been a decrease in the number of students who drop out from the course towards the end of a semester for having done too little work during the first months. Furthermore, even the leading students of the course admit that they have plenty of food for brain racking in the tests.

From the teacher's perspective, the tests provide us with a means allowing us to secure, without much extra work, that the students are familiar with the material of the previous lecture before they start to learn a new one. On the other hand, it is a means that allows us to obtain feedback on the learning of both individual questions and comprehensive themes. The investigation of the answers of current tests has allowed us to add an item on the spot or explain a question that has remained obscure. A fairly efficient means of discovering the most difficult themes of the entire course is an analysis of incorrect answers found in the summary tests of about twenty most successful students. The most difficult theme of the tests was the set of questions added in the last year for evaluating the suitability of possible steps in concrete proof-building situations (see 4.8). Even better students made fairly many mistakes here, which did not disappear even in the summary test. Questions on the comparison of formulas in predicate logic also proved difficult (Figure 3). Besides the themes posing difficulties, the summary tests also reveal occasional weaker answers that point to certain gaps in lectures/study aids or to differences in the approach of different workshop groups.

The themes and the organization of tests have steadied; we have a bank of questions for generating a reasonable number of parallel variants. To discourage the mechanical solving of tests, the author is currently considering the imposition of a limit to the number of tries or the linking of the effective score to the number of the try. However, we do not consider taking very complex measures. Even in their current form, the tests provide students with enough opportunities and motivation for reasonable work.

REFERENCES

- Croft A.C., Danson M., Dawson B.R., Ward J.P., 2001, "Experiences of using computer assisted assessment in engineering mathematics", *Computers & Education, **37**,* 53-66
- Prank, R., 1991, "Using Computerised Exercises on Mathematical Logic". *Informatik-Fachberichte*, vol. 292, Berlin: Springer, 34-38.