Some Further Steps after the Computerisation of Predicate Calculus

Rein Prank
Tartu University, Estonia
prank@cs.ut.ee

Abstract. In Tartu University the exercises in Mathematical Logic were computerised about ten years ago. The paper investigates the changes in the course induced by Computerised Teaching of Formal Proofs. In many cases we have now included the references to the students' practical experience and it has improved the understanding of the theory. We have rebuilt the Chapter of Predicate Logic justifying the 'natural character' of the rules of Formal Calculus. We try to demonstrate the use of Logic and Formal Rules in finding the proofs of 'real' theorems.

1. Our starting point in 1987 and the programs

At the end of 1987 we started a teaching software project for the exercises in Mathematical Logic at Tartu University. At that time our university followed a typical Soviet curriculum and our first-year students of Mathematics and Computer Science had the compulsory course with the following content at the second term:

I. Propositional Logic. *Introduction. Sentences, truth-values, connections, formulas, truth-tables. Tautologies, logical equivalence. Expressibility by {&,Ø}, {Ú,Ø}, {É,Ø}. Normal forms.*
II. Predicate Logic. *Predicates, quantifiers. Validation of formulas on N, Z, R, P(N). Writing the predicates and theorems by formulas. Signature, first-order language, theory of a model. Tautologies. Logical equivalence. (++1). (++2). Prenex form.*
III. Axiomatic Theories. *Axioms and rules of Propositional Calculus. Examples of the proofs. Consistence and completeness of Propositional Calculus. Predicate Calculus. Examples of the proofs. First-Order Theories, Formal Arithmetic. (++3)*
IV. Algorithm Theory. *Turing Machine. Computing N®N functions on TM. Enumeration of TM. Halting Problem. Overview of decidability results. Their meaning for programming.*
Now this course is moved to the second year of the studies and it is possible to add some material. The symbols *(++1)-(++3)* denote the additions described in the present paper.

The decision to use the computers was initially caused by poor performance of the students in two subject areas:
1) programming Turing Machines,
2) construction of the proofs in formal calculi.

In 1988-90 we implemented two programs TURING MACHINE and PROOF EDITOR for those most problematic exercises and in addition two other programs for Chapter I of the course:
3) TRUTH-TABLE CHECKER (exercises on filling the truth-table, checking tautologicity and equivalence of the formulas, finding the formula for a given Boolean function),
4) ALGEBRAIC MANIPULATION ASSISTANT (exercises on expressibility by {&,¬}, {∨,¬}, {⊃,¬} and on normal forms.

The programs covered the most technical part of the exercises of three chapters. Teaching of Model Theory of Predicate Logic was continued in a traditional way. From 1991 we have exploited our software with enrolments of 70-100 students. The first overview of our computerised exercises is given in [3].
Probably the most original part of our software constitutes the two programs concerning Model Theory of Propositional Logic. Different features of those programs and the lessons learned are described in [4], [5] and [6]. The short paper [8] describes our problem solving tests using all four programs. But in the present paper we consider only the proofs. We try to examine the changes in the course caused by the fact that the students construct a large number of formal proofs using the computer.

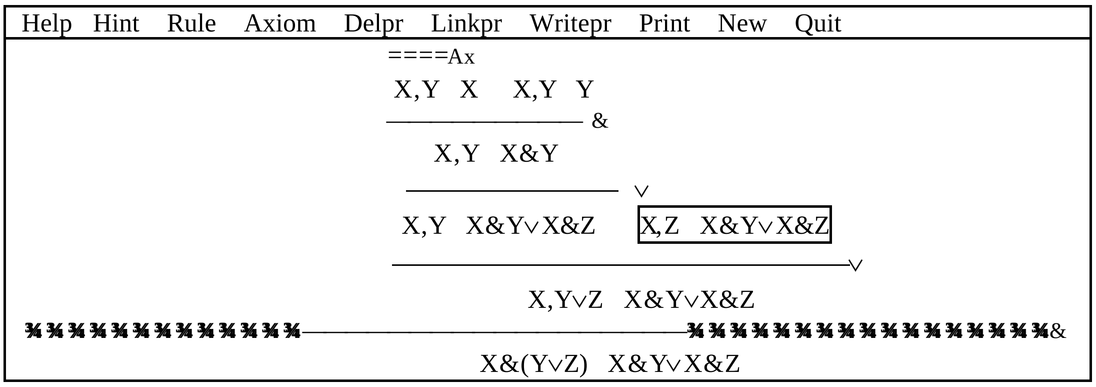
2. Our exercises with PROOF EDITOR

Already several years before the computerisation we had taught Gentzen type Sequential Calculus [2] with the sequents of form *A* (just one formula in succedent), with axioms *,A, A* and with logical rules (there are usual restrictions for *" and $*):

```
   A;      B      ,A,B  C         A           B        ,A  C;    ,B  C       ,A  B          A;      AÉB
¾¾¾¾¾¾  ¾¾¾¾¾     ¾¾¾¾     ¾¾¾¾    ¾¾¾¾¾¾¾¾¾¾  ¾¾¾¾¾    ¾¾¾¾¾¾¾¾¾¾
    A&B         ,A&B  C        AÚB          AÚB         ,AÚB  C            AÉB                 B

 ,A  B;     ,A  ØB        ØØA           A(x)          ,A(t)  B          A(t)            ,A(x)  B
¾¾¾¾¾¾¾¾¾¾¾¾¾¾    ¾¾¾¾¾¾¾¾¾¾¾¾¾
    ØA                      A           "xA(x)       ,"xA(x)  B        $xA(x)          ,$xA(x)  B
```

Quite a similar system is used in [1]. We have chosen this system mainly because of three reasons. The first is the simplicity of finding the proofs. The second reason is that all the rules are familiar to the students from real proofs. Further, the tree form of the proofs sounds well with other disciplines of Computer Science. But trying to keep the connections with other mathematical subjects we did not want to introduce the sequents with more than one formula in succedent.
The screen of the PROOF EDITOR consists of two windows. The Rule Window contains the (above) logical and the structural rules. The proofs are built in the Proof Window from the root upwards until all the leaves are axioms. At any step the student has to select an inference rule from the Rule Window. If it is possible to apply the selected rule, then the upper sequent(s) of the rule is (are) added to the tree. If the upper part of the rule contains a new formula (or term), then the student has to enter it. The figure below presents the Proof Window. The box marks the active sequent.



```
Help   Hint   Rule   Axiom   Delpr   Linkpr   Writepr   Print   New   Quit
                                      ====Ax
                                       X,Y  X     X,Y  Y
                                      ———————————— &
                                          X,Y  X&Y
                                        ———————————— ∨
                                        X,Y  X&Y∨X&Z       X,Z  X&Y∨X&Z
                                       ———————————————————————————— ∨
                                                  X,Y∨Z  X&Y∨X&Z
¾¾¾¾¾¾¾¾¾¾¾¾¾——————————————————¾¾¾¾¾¾¾¾¾¾¾¾¾¾¾¾¾¾¾&
                                              X&(Y∨Z)  X&Y∨X&Z
```

PROOF EDITOR contains a propositional prover that works in the same system. The prover is used for two purposes:

1) for dynamic hints (what rule to use and what formula to enter),
2) for the comparison of the size of the student's proof with the 'ideal' proof.

The program is not able to solve the tasks in Predicate Calculus but only checks the correctness of the proof. We worked with PROOF EDITOR successfully several years on the IBM PC AT 286 computers. The description of the program is given in [7].
After some years of practice the compulsory exercises stabilised on nearly 40+40 proofs in Propositional and Predicate Calculus. The main part of our tasks is received from the standard equivalences of Propositional and Predicate Logic, splitting equivalence into two sequents. Such collection of tasks gives the students quite a massive experience and they have the possibility to see the properties of all connections and quantifiers. But we have not set the goal to teach average students to prove hard theorems.

3. Computerisation and Learning of the Theory

We use the computers to cope with some bottlenecks in understanding the subject. But after some time the new technology itself causes some changes in the content and ideology of the course.

The first result of computerisation of formal proofs was that we established a normal ratio between the work of the teacher and the work of the student in this chapter. Probably the ability to build the proofs is something quite important for the students. They work enthusiastically and learn more than earlier. Their enthusiasm gives us the possibility to turn their interest to the learning of theoretical issues as well, if the lecturer creates the connections between the lectures and the new practical experience. Some examples:

1) Correctness Theorem states that every provable sequent is a tautology. But some of the rules enable to get upper sequents that are not tautological and cause the deadlock in building of the proof. We can demonstrate how the theory is related to the success of the students on the tasks.

2) If the student builds an unprovable sequent inside a propositional task and selects the rule for the next step before deleting the wrong node, then he gets a small penalty in the score. But the program is not able to find the same error in predicate exercises. After some time the students understand that a decidable case with penalties is better than an undecidable case without them.

3) In case of Propositional Calculus the program checks (by calculation of truth-table) the tautologicity of newly added upper sequents. Here the students have the possibility to try the formulas with the increasing number of variables and can study the impact of exponential complexity.

4) At some moment the students ask the question: what is necessary for being able to build the proofs for arbitrary provable sequent? The proof of the completeness theorem for Propositional Calculus is constructive and we can try to extract the answer from the proof.

4. Before and after teaching of Formal Calculus

If the formal system is only an object of theoretical investigation, then the student has quite a limited interest to its axioms and rules. Many textbooks reflect this, giving only the proofs of soundness and completeness of the whole system. If the student has to build the proofs, then the properties of particular axioms and rules (as well as the system as whole) will determine the necessary way of thinking and acting of the student. The student asks why the system contains just those axioms and rules. Where are they from? The Calculus we choose for teaching can support the earlier proof skills or it can generate the idea that the proofs in Logic are something totally different from the proofs in ‘real Mathematics’.

We prepare the introduction of the formal system already in Chapter II. This chapter contains a usual list of the main predicate equivalences. The proof of every equivalence consists of the proofs of two sequents. The lecturer underlines that every step in the proofs is done using some (known already earlier for the students) standard rule. All these rules are applied to the main connective of some formula (the formula to be proved or the formula known to be true). So the rules describe how to prove the conjunction, the disjunction etc and how to use the conjunction, disjunction etc in the proof. After completing the equivalences our next lecture (denoted by *++1* in the above course description) formulates these two rules for every connective. The same rules form the Predicate Calculus of Chapter III. After extracting the rules we use them proving some small theorems from ‘real Mathematics’ (*++2* in Section 1, some examples are in Section 4). The result is that in Chapter III the formulation of Propositional and Predicate Calculus is for the students some formalisation of existing proof technique.

We continue this topic after teaching the Formal Predicate Calculus with necessary amount of practical exercises. Now the students understand much better the anatomy of the proofs and it is possible to describe the bridge between Logic and Mathematics. We try to demonstrate that the rules of Formal Logic can help to find the proofs. If we express the theorem as a sequent, then the form of the formulas in antecedent and succedent predicts us the logical steps in the proof. In many proofs in University Mathematics only one or two steps contain some creative act. The remaining part is only some routine application of the rules. The ability to build the proofs has two components: the technical skills and the creativity. We try to convince the students that at least one of the components (the Logic) can be learned. In the example proofs we demonstrate the almost algorithmic character of finding the proof and differentiate the purely logical steps and creative steps. Good examples for demonstration are the following theorems:

1) If *R(x,y)* is an order relation then *R(x,y)&Ø(x=y)* is a strict order relation,
2) If *R(x,y)* is a strict order relation then *R(x,y)Úx=y* is an order relation,
3) *"x"y$z (x∗z=y)* in Group Theory,
4) Equivalence of the axioms in the terms of order and in the terms of operations for Lattices and Boolean Algebras,
5) Associativity, distributivity and commutativity laws in Peano Arithmetic.

We can collect more such examples but we have no time to use them if we do it on the blackboard. A real challenge for the creators of the Tools is to create a syntactically flexible PROOF EDITOR for adopting the examples from different areas of Mathematics. Such a program should enable us to use the axioms, the ‘earlier’ theorems (given together with the task), the previous items of the same collection of the tasks. Very important is the convenient implementation of replacing the term by an equal term. We have used (denoted above by *++3*) such computerised exercises for Peano Arithmetic but our program is implemented in DOS text mode and is already quite old.

References

[1] Jershov, Yu.L. and Palyutin J.A. *Mathematical Logic*. Nauka (Moscow), 1979.

[2] Kolmogorov, A.N. and Dragalin, A.G. *Introduction to Mathematical Logic (Russian).* Moscow University Press, 1982.

[3] Prank, R. Using Computerised Exercises on Mathematical Logic. Informatik und Schule 1991. *Informatik-Fachberichte,* 292, 34-38. Springer-Verlag, 1991.

[4] Prank, R. and Viira, H. Algebraic Manipulation Assistant for Propositional Logic. *Computerised Logic Teaching Bulletin*, Vol.4, No.1, St Andrews Univ., 1991, pp. 13-18.

[5] Prank, R. Towards Flexible Programs for Exercises in Mathematics. *Hypermedia in Tallinn'96. May 22-24, 1996. Proceedings.* Tallinn Technical University, 1996, 188-192.

[6] Prank, R. Good diagnostics = adequate stepwise solution interface? *Proc. International Conference on Technology in Mathematics Teaching.* Koblenz, 1997, CD- and WWW-publication (http://euler.uni-koblenz.de/). 9 pp.

[7] Prank, R. and Ounapuu, T. Intelligent Tutoring System for Proof Building Exercises. *New Media and Telematic Technologies for Education in Eastern European Countries.* Twente University Press, Enschede, 1997, 371-376.

[8] Prank, R. Using Computers for Problem Solving Tests. *International Conference on the Teaching of Mathematics. Samos, Greece, July 3-6, 1998*. John Wiley & Sons, Inc. Publishers, 1998, 248-250.