Computers in Education, O. Lecarme and R. Lewis (eds.)

COMPUTER-ASSISTED AXIOMATIC MATHEMATICS: INFORMAL RIGOR

R. L. Smith, W. H. Graves, L. H. Blaine, V. G. Marinov

Institute for Mathematical Studies in the Social Sciences
Stanford University
Stanford, California, U.S.A.

Teaching axiomatic mathematics on a computer has traditionally involved the construction of derivations in a formal system. Heretofore these formal systems have simply been adopted from mathematical logic. Such systems were designed, however, to make it easy to prove results about the system rather than to make it easy to prove results in the system. This paper describes a computer program for teaching axiomatic set theory that makes use of complex inferential methods that are informal and semantically transparent to the user while remaining fully rigorous. This interactive proof checker permits the student to submit informal proofs to be checked for correctness by the program, thus combining the clarity of an informal approach with the rigor of a formal one.

1. INTRODUCTION

This paper describes the current state of a CAI system used for teaching axiomatic mathematics under development at the Institute for Mathematical Studies in the Social Sciences at Stanford University. We feel that by the fall of 1975 our system will embody the current state of the art in computer science in its use of computer audio, CRT displays, natural-language processing, theorem proving, and proof checking.

Our primary goal is an interactive computer system that allows for the rigorous development of formal mathematical systems while presenting the formal systems with more natural language and conducting user/computer dialogue at the natural-language level rather than at the level associated with fully explicit, completely detailed formal proofs.

Proofs, as usually presented in lectures and textbooks, are informal sketches rather than formal proofs in the traditional sense of formal axiomatic systems, as complete formality would be too tedious, time consuming, and unclear. It is generally believed that such informal proofs can be translated into formal proofs in formal systems if necessary. Our system does not effect such a translation, though such a translation is implicit. Rather, we have made precise some of the derivation methods associated with informal proofs. This allows students to enter informal proofs which are checked for logical correctness by machine, achieving the clarity of informal presentation with the rigor of a formal approach. It is a common, but serious, error to believe that such an informal approach is not rigorous. It is precisely the ability to recognize and omit from explicit mention the inconsequential steps of an argument that characterizes mathematically mature rigor.

This latter conception of mathematical rigor is implemented in a production CAI system named QUIP used at Stanford University in the fall quarter, 1974, to teach Philosophy 161, "Introduction to Set Theory." The course is for upper-level undergraduate students and traditionally presents axiomatic set theory as developed in Suppes (1972). Set theory is a good initial curriculum for several reasons. Much of modern mathematics is founded on set theory. The cumulative development of a mathematical theory is crucial to its implementation on a machine; set theory has a uniform development, beginning with simple results and working up to results that are as complex as any other part of mathematics. Having this accumulation of concepts, theorems, and proof techniques is a prerequisite to computer proof checking in mathematics beyond elementary logic. Further, the number of proofs that the student does in the curriculum is large and the computer is able to check them much more carefully than a classroom teacher would be able to do; this presents the potential for an immediate gain for the student.

2. THE LANGUAGE OF SETS

The language of the first-order predicate calculus, with additions for set theory, readily lends itself to machine implementation. It is a simple example of a context-free language. The symbols or vocabulary of the language are easily specified, as are the rules of formation. Internally, formulas are stored as LISP s-expressions written in Polish notation. For example, the formula

$$(\forall x)(x \in A \vee x \notin B) \qquad (1)$$

is stored internally as

$$\begin{array}{l}(\text{UNIV } (x)\ (\text{ORSGN } (\text{IN } x\ A)\\ \quad (\text{NEGSGN } (\text{IN } x\ B)))) .\end{array} \qquad (1^*)$$

However, it is hardly sufficient to recognize only such formulas as formula (1) above. The language in which set reasoning is ordinarily discussed is richer--so much so that it approaches the complexity of informal English. We examined several texts to find the largest habitable language that could readily be parsed by a context-free grammar with semantic functions associated with the rules of the grammar producing the internal representation (see N. Smith, 1974, and R. Smith, 1974). The grammar of QUIP requires explicit mention of variables and their scopes, and does not allow for anaphoric reference of any reference to the

BOOLE.[4] BOOLE is made available to the student after he has proved enough of the standard results to get a good sense of the class of theorems that are involved in this "bag." The last of the above facts of quantifier-free Boolean algebra is one of the elementary results appealed to in the proof of the Schröder-Bernstein theorem.

Additional methods that we think will be useful in set theory include the (full) logic of equality, the elementary theory of relations and Cartesian products, the theory of finite sets, and the elementary algebra of the cardinal and ordinal numbers. We are currently working on adding these to QUIP.

6. THE OBVIOUS STEP

A standard feature of sophisticated mathematics is an appeal to the "obviousness" of a given step. What is an obvious step depends on the mathematical level of the subject matter. For example, in set-theoretical proofs a series of logical manipulations are usually compressed into one step, whereas in real analysis proofs a series of set-theoretical manipulations may be considered obvious and taken in one step. The above is also true with a single subject. Working on the general developments of set theory one might want to prove the commutativity of the union operation, whereas in the treatment of equipollence or cardinal numbers, replacement based on this property of sets is a very simple step indeed.

In Suppes (1972) one finds inferences for which the justification is "by sentential logic" or "by quantifier logic." When a step is justified by sentential logic the task is simple, because it will only require the TAUTOLOGY rule.[5] Because of the decidability of sentential logic the inference will always be confirmed.

A step following by quantifier logic makes matters considerably worse. Sometimes such inferences, which to human insight seem comparable to or even simpler than a similar tautological inference, might be quite hard to prove mechanically. Our current approach is to use a resolution theorem prover with equality replacement employing the MU-strategy (Marinov, 1973). The prover is used at the present time mainly for the rules VERIFY and CONTRADICTION.

The user is not allowed to intervene in the resolution process except for asking certain inferences to be verified.[6] The problem here is how to "tune" the prover in order to optimize the number of inferences it obtains. Completeness is restricted severely in many different ways. From the user's point of view it makes no difference whether the prover does not detect a valid inference because the limitations of time and space prevent a complete strategy or because the inference is blocked by some restriction. Furthermore, it is important that the prover stops early when asked to VERIFY an invalid inference.

Our experience so far is that, contrary to some current beliefs, a properly organized resolution theorem prover can be useful in proof checking. It is possible to have a small but efficient prover which gets most of the needed inferences. One disadvantage ought to be mentioned, however. There is some discrepancy in the human user's sense of an obvious inference and what the prover will accomplish. Thus, while some steps not even clear to the user may be accepted by VERIFY, it may fail at steps which the user considers obvious.

7. THE STRUCTURE OF PROOFS

We have been discussing the machinery that makes proofs shorter and more compact. The important use of these methods, however, is to allow the proofs to take on a perspicuous structure. Consider the following example of an informal proof, as it might be seen in a textbook, followed by the corresponding QUIP proof.

The theorem is the classical result that the Russell set--the set of all x such that x is not in x--is empty.[7] The textbook proof goes as follows:

> Let R be the set of all x such that x is not in x. Suppose that R is not empty. Then by Theorem 1.9 we have
>
> for all x $x \in R$ if and only if $x \in x$
> and, hence, $R \in R$ if and only if $R \notin R$
>
> which is a contradiction. QED

This proof is expressible in QUIP as follows:

```
Derive:  {x: x ∉ x} = 0

ABBREVIATION
        (1) R = {x: x ∉ x}
ASSUME  (2) R ≠ 0
(2 IMPLICATION using Theorem 1.9) US
        (3) R ∈ R iff R ∉ R
3 CONTRADICTION
        (4) R = 0
QED
```

The QUIP proof given above uses internally some reasonably complicated algorithms in order to let the student's proof preserve the (informal) linguistic and proof-theoretic structure of the textbook derivation. The steps of the proof are: (1) the student makes an ABBREVIATION which allows him to type R and mean the Russell set; (2) ASSUME makes a temporary assumption as part of the indirect proof strategy; (3) the student uses the IMPLICATION rule, which makes the obvious instantiation into Theorem 1.9 (previously proved); and (4) the CONTRADICTION rule is used, which calls the resolution theorem prover to see that line 3 is a simple contradiction. Note that

the student compresses two steps of the textbook proof into line 3.

Often a single English sentence is used to express a step of an argument that involves many separate axioms, definitions, theorems, and rules of inference. Of course, this involves many subtle problems of natural-language processing. Currently we have designed a mechanism whereby complex commands can be given and executed by QUIP. This EVALSTEP mechanism is similar to LISP EVAL in that it is an inside-out recursive evaluation. Differences include the following: (1) the handling of several kinds of global information about the proof is more complex than EVAL; (2) when insufficient information is given, EVALSTEP must suspend evaluation and obtain more information, either from the user or from the data structure currently maintained about the proof; (3) when errors occur, it is important to back up the command to the most recent point from which the command can sensibly continue, with interactive correction being obtained.[8]

The following is an example of a proof that uses complex commands. Note that the syntax currently implemented is not satisfactory since it is too unnatural for student use. However, it is a step in the direction of natural-language proofs, because the machinery used here is the correct control structure for evaluating multiple commands when the planned grammatical mechanism is added in early 1975.

In the following example, student (or user) input is underlined, and comments are in italic.

Derive: $A = \{x:x \in A\}$

((AXIOM OF SEPARATION)INSTANTIATE)REPLACE FORMULA

[Note: *The informal syntax corresponding to this complex command is: Replacing* x *in* A & x *in* A *by* x *in* A *in the result of putting* x *in* A *for* FM(x) *in the axiom schema of separation we have that* (1) $(\exists C)(\forall x)(x \in C \leftrightarrow x \in A)$.]

Schema: $(\forall B)(\exists C)(\forall x)(x \in C \leftrightarrow x \in B \ \& \ FM(x))$
What shall we put for FM *↑P

[Note: *The ↑P is a request to the checker to print out the pending (i.e., current) step in the evaluation of the complex command.*]

The pending step is:
Axiom Schema of Separation

$(\forall B)(\exists C)(\forall x)(x \in C \leftrightarrow x \in B \ \& \ FM(x))$

What shall we put for FM *$x \in A$
Designated variable of FM *x

[Note: *The designated variable of* FM *is the free variable of* FM *to be bound by the* $(\forall x)$ *quantifier of the schema.*]

INSTANTIATE *B
INSTANTIATE for B *A
Replace *↑P

The pending step is:

Replace formula into

$(\exists C)(\forall x)(x \in C \leftrightarrow x \in A \in x \in A$

Replace *$x \in A \ \& \ x \in A$
By *$x \in A$
Using *↑P

The pending step is:

Need a justification for replacing

$x \in A \ \& \ x \in A$

by

$x \in A$

Using *TAUTOLOGY

(1) $(\exists C)(\forall x)(x \in C \leftrightarrow x \in A)$

1 IMPLIES USING * THEOREM * ABSTRACTION
(2) $(\forall x)(x \in \{x:x \in A\}$ iff $x \in A)$
2 IMPLIES USING * AXIOM * EXTENSIONALITY
(3) $\{x:x \in A\} = A$
QED

8. THE SCHRÖDER-BERNSTEIN THEOREM--A SAMPLE PROOF

As an illustration of a complex proof in QUIP we offer the Schröder-Bernstein theorem. One classical proof of the theorem is to construct a bijective function between two sets B and C from the functions F and F1 that are given immediately from the hypothesis of the theorem. The proof given below, which follows Suppes (1972, p. 95), uses several previously stated theorems, axioms, and definitions. Instances of the VERIFY, BOOLE, and IMPLICATION rules are included. One theorem (3.5.7) is ordinarily a part of the Schröder-Bernstein theorem, but we made it a separate theorem in our curriculum. The proof of Theorem 3.5.7 is approximately 15 lines long.

We use the notation F:B INJ C and F:B BIJ C for, respectively, "F is an injection from B into C" and "F is a bijection from B onto C." The image of a set D under a function F is written as F"D, the restriction of a function F to a set D is written as F\D, and $\breve{F}$ is written for the converse of a function. The numbered lines of the derivation follow the justification (or command) for the line.

Derive: $B \preccurlyeq C \ \& \ C \preccurlyeq B \rightarrow B \approx C$

HYP (1) $B \preccurlyeq C \ \& \ C \preccurlyeq B$
(1, Definition LEQUIPOLLENCE, VERIFY) ES
(2) F:B INJ C & F1:C INJ B
(2, Definition INJECTION, Theorem 3.7.5, VERIFY) ES
(3) $D \subseteq B \ \& \ F1"(C - (\breve{F}"D)) = B - D$

```
(2 SIMP 1),(3 SIMP 1), Theorem 3.5.18 IMPLIES
     (4) F\D:D INJ C
4, Theorem 3.5.20, IMPLIES
     (5) F\D:D BIJ (F\D)"D
((BOOLE: D ⊆ D), Theorem 3.1.32 IMPLIES) US
     (6) (F\D)"D = F"D
5 RE 1 Using Line 6
     (7) F\D:D BIJ F"D
(2 SIMP 2), Theorem 3.5.13 IMPLIES
     (8) F̆1:F1"C INJ C
ABBREVIATION
     (9) E1 = F̆1\(B - D)
((BOOLE: C - (F"D) ⊆ C), Theorem 3.1.34
   IMPLIES) US
     (10) F1"(C - (F"D)) ⊆ F1"C
3,8,10 Theorem 3.5.16, Theorem 3.5.20 VERIFY
     (11) E1:B - D BIJ E1"(B - D)
((BOOLE: B - D ⊆ B - D), Theorem 3.1.32
   IMPLIES) US
     (12) E1"(B - D) = F̆1"(B - D)
ABBREVIATION
     (13) E2 = C - (F"D)
(BOOLE: F̆1"(F1"(C - (F"D))) = C - (F"D)),2,
   Theorem 3.5.14 VERIFY
     (14) F̆1"(F1"E2) = E2
3,11,12,14 VERIFY
     (15) E1:B - D BIJ E2
BOOLE
     (16) D ∩ (B - D) = 0 & (F"D) ∩ E2 = 0
7,15,16, Theorem 3.5.21 IMPLIES
     (17) (F\D) ∪ E1:D ∪ (B - D) BIJ (F"D) ∪ E2
3 BOOLE
     (18) D ∪ (B - D) = B
(Definition IMAGE) US
     (19) F"D = RNG(F\D)
2,3, Theorem 3.5.7, Definition INJECTION VERIFY
     (20) F\D:D → C
20, Definition MAP VERIFY
     (21) RNG(F\D) ⊆ C
19, 21 BOOLE
     (22) (F"D) ∪ E2 = C
17 RE Using Line 18
     (23) (F\D) ∪ E1:B BIJ (F"D) ∪ E2
23 RE Using Line 22
     (24) (F\D) ∪ E1:B BIJ C
24, Definition EQUIPOLLENCE VERIFY
     (25) B ≈ C
1,25 Conditional Proof
     (26) B ≼ C & C ≼ B → B ≈ C
QED
```

The next steps that we will take include implementing an informal proof grammar to replace the awkward commands in the above proof, additional proof procedures to handle more complex substitutions and replacements, the use of metalanguage to describe inferences and to state metatheorems, and metatheoretic proof procedures for proving metatheorems.

9. CONCLUSION

The importance of the project is that it shows how axiomatic mathematics can be handled in an informal and natural way on a computer. We have stressed here the application of this approach to instruction. We should add the prediction that mathematicians will come to use interactive proof checkers in a way analogous to the present use of computers by mathematicians.

Concerning the relation of humans to computers, it has become fashionable to regard informal reasoning and mathematical logic as incompatible. The mistake is to assume that a logical treatment necessarily implies a completely detailed and atomic treatment of every step. The QUIP system shows that this is not necessary in that the steps of an argument can be quite large and can rely on implicit information, and that the argument can be informally stated. What emerges from this system of reasoning methods are proofs that bear a stronger relation to informal arguments than would be possible in a classical formal system as usually understood.

10. NOTES

We would like to acknowledge our debt to Professor Patrick Suppes for his direction and assistance in this project, and for the facilities of the Institute for Mathematical Studies in the Social Sciences at Stanford University. Also we thank Edward Bolton, Larry Markosian, Inge Larsen, Ronny van den Heuvel, and David Ferris for their assistance. The research reported here was supported by the National Science Foundation under NSF Grant EC-43997 for research into natural language processing in computer-assisted instruction.

[1] This syntactic principle is known as the lemma of alphabetic invariance.

[2] For the justification of this rule see Church, 1956, p. 94. The implementation of this rule in QUIP uses a small compiler that compiles PDP-10 machine code that computes the value of the 2^n lines of the truth table (where n is the number of distinct atomic sentences found in the formula). We are grateful to R. Weyhrauch for this idea. Our algorithm differs from his in being faster for smaller formulas and slower for larger ones.

[3] In the logic course taught at Stanford using QUIP, the tautology rule is given to the students after they have dealt with inferences in the propositional calculus and have been exposed to truth tables and sentential validity.

[4] We thank John McCarthy for the suggestion of the BOOLE rule.

[5] That is, provided the appropriate formula replacements are possible.

[6] Human intervention in the resolution process has generally involved manipulation of certain unintuitive parameters guiding the search.

[7]The Russell set R is the set of all sets that are not elements of themselves. As Bertrand Russell (1967) first showed in 1902, this cannot be a well-defined set without generating a contradiction. Various set theories handle this in different ways. In Gödel-Bernays set theory, the Russell "set" is a proper class, considered too "big" to be a set. In Suppes' (1972) formulation of Zermelo-Fraenkel set theory, such objects are sets but they are empty. This accounts for the statement we give of the theorem.

[8]We have examined some of the features of INTERLISP (Teitelman, 1974) in regard to the way it allows debugging of a user's LISP program. Unfortunately, most of the mechanisms provided there require understanding of the LISP language and implementation. Our goal is to have a system that can be understood without any appreciation of its computer implementation.

11. REFERENCES

Church, A. (1956), Introduction to Mathematical Logic, Princeton University Press, Princeton, N. J.

Kleene, S. C. (1952), Introduction to Metamathematics, Van Nostrand, New York.

Marinov, V. G. (1973), Maximal Clause Length Resolution, Doctoral Dissertation, University of Texas, Austin, Tex.

Quine, W. V. O. (1959), Methods of Logic, Wiley, New York.

Russell, B. (1967), "Letter to Frege," in van Heijenoort (ed.), From Frege to Gödel, A Source Book in Mathematical Logic, 1879-1931, Harvard University Press, Cambridge, Mass.

Smith, N. (1974), A Question-Answering System for Elementary Mathematics (Tech. Rep. No. 227), Institute for Mathematical Studies in the Social Sciences, Stanford University, Stanford, Calif.

Smith, R. L., Smith, N. W., and Rawson, F. L. (1974), CONSTRUCT: In Search of a Theory of Meaning (Tech. Rep. No. 238), Institute for Mathematical Studies in the Social Sciences, Stanford University, Stanford, Calif.

Suppes, P. (1957), Introduction to Logic, Van Nostrand, Princeton, N. J.

Suppes, P. (1972), Axiomatic Set Theory, Dover, New York.

Teitelman, W. (1974), INTERLISP Reference Manual, Xerox, Palo Alto, Calif.

Winograd, T. (1971), Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, Doctoral Dissertation, Massachusetts Institute of Technology, Cambridge, Mass.