Robert S. Boyer
Computer Sciences Department
University of Texas at Austin

Statement on Receipt of AMS Prize in Automatic Theorem Proving

Like chess and checkers, mathematics is a game that can be played by a computer. As Gödel remarked, "The development of mathematics toward greater precision has led, as is well known, to the formalization of large tracts of it, so that one can prove any theorem using nothing but a few mechanical rules." In chess and checkers, some computers play better than most experts, but this is far from true, today, about computers in mathematics. Only in a very few areas of mathematics, such as in the factorization of large integers, symbolic integration, and a few parts of geometry are computers as good as expert mathematicians. Why is mathematics seemingly so much harder than chess or checkers to mechanize successfully? One reason is that in mathematical proofs, there is, at any point, an *infinite* number of moves one can make. (For example, one could choose to instantiate the theorem that addition is commutative with arbitrarily large integers.) The experience that there is an infinite number of legal moves that one could make in a proof is a common experience: think of all the proofs of deep theorems whose crucial step is something like "consider the following set," where the set in question is not something one would have considered for years.

The Current Prize in Automatic Theorem Proving, which I am so pleased to share with J Moore, my colleague of almost twenty years, has been awarded to us for the construction and application of a computer program that proves theorems in a variant of elementary number theory. (The "variation" is to include enough camouflage in the form of datatypes such as ordered pairs and strings as to give the theory a computer science feel, akin to that of Pure Lisp, and a practical bent.) For the most part, the prover is constructed on principles of automatic theorem proving not discovered by us but rather by many people to whom we are greatly indebted (e.g., Martin Davis, Hao Wang, A. Newell, H. Simon, J. McCarthy, R. Burstall, J. A. Robinson, L. Wos, D. Loveland, R. Kowalski, W. W. Bledsoe, D. Knuth, L. Hodes, and M. Ballantyne). Not to suggest that it was easy to orchestrate the good ideas we have borrowed from so many brilliant fellows! Perhaps the most original contribution we have made to the science of automatic theorem proving is in our implementation of a method for guessing which induction to try. Because first-order elementary number theory has an infinite number of instances of the induction scheme as axioms, deciding which of these axioms to consider is a nontrivial problem. Very roughly speaking, our "heuristic" for choosing an induction is to try is to use the dual of the recursive definitions of the concepts involved in the conjecture we are trying to prove. The details of our prover can be found in our books *A Computational Logic* and *A Computational Logic Handbook* (Boyer & Moore, Academic Press, 1979 and 1988).

The successful application of our prover to thousands of theorems has been undertaken by many of our colleagues, to whom we are most grateful. The many expert users of our system merit having jointly received the Current Prize with us; particularly noteworthy are Bill Bevier, Bishop Brock, John Cowles, Ben DiVito, Art Flatau, David Goldschlag, Don Good, C. H. Huang, Chris Lengauer, Warren Hunt, Matt Kaufmann, David Russinoff, N. Shankar, Ann Siebert, Matt Wilding, Bill Young, and Yuan Yu. Pointers to the amazing number of theorems they have induced our prover to check can be found in "A Theorem Prover for a Computational Logic" (Lecture Notes in Computer Science 449, Springer-Verlag, 1990, pp. 1–15.) The success of the prover in checking all those theorems, many of which are nontrivial, e.g., quadratic reciprocity and the Church-Rosser theorem, is primarily due to the persistence of the users of our prover in guiding it towards proofs. What are the principal means that the users use to guide the prover? One means is to state intermediate lemmas. These lemmas act almost like logic programs in guiding the prover to consider certain subgoals. For example, a lemma of the form

$$p(x) \wedge q(x) \rightarrow f(x) = g(x),$$

once stated and proved, would be a hint to the prover to attempt to rewrite any term $f(t)$ later encountered to the term $g(t)$, provided it is possible first to establish the subgoals $p(t)$ and $q(t)$. Another means is to define recursive concepts in such a way that our prover will be tricked into doing the right induction!

Because progress in automatic theorem proving is slowed by the paucity of mathematicians who work in the field, there are two points that I want to make. First, checking, as opposed to discovering, mathematics seems technologically feasible with available computing technology. Not only our prover but those of de Bruijn, Suppes, Bledsoe, Constable, McCune-Lusk-Overbeek-Wos, Gordon-Milner-Morris-Newey-Wadsworth, Farmer-Guttman-Thayer, and others have been used to check a wide subset of elementary mathematics, and there seems hardly any barrier to checking any part of it for which one has the patience. As a rule of thumb, it may be guessed that checking a result by machine (for a skilled user) will take no more than an order of magnitude longer than it would take to write out the proofs (and all antecedent foundations) at the level of detail of an undergraduate textbook. Is not the evidence of the feasibility of mechanically *checking* proofs also some evidence that the time is ripe for further progress in mechanically *discovering* proofs, by talented mathematicians? Second, it should perhaps be revealed that in addition to possible economic benefits in realms such as mechanical program verification, building

and using a theorem prover can be as aesthetically satisfying as playing chess or chamber music. We are most grateful to the people who have sponsored our research (Tom Keenan, Bob Grafton, Ralph Wachter, Bill Scherlis, K. Abdali, and others at ONR, NSF, DARPA and elsewhere).

Biography

Robert S. Boyer was born August 2, 1946 to Fred Y. Boyer and Catherine A. Boyer. Prior to his birth, his mother had been cracking coded messages while his father was battling kamikazes. Prior to college, Boyer attended eleven schools as his father practiced law in naval bases up and down the coasts. Boyer has been married to Anne Olivia Herrington since 1966, and they have three children: Madeleine, Margaret, and Nathaniel.

As a high school student in Kingsville, Texas, Boyer made a presentation to a Texas meeting of the AMS on some theorems about three dimensional matrices and attended the NSF John von Neumann summer math institute at SMU. As an undergraduate at the University of Texas at Austin, Boyer studied math under R. L. Moore and H. S. Wall; logic under Paul Lorenzen and Norman Martin; and philosophy under John Silber and Tom Gould. As a mathematics graduate student at the University of Texas at Austin, Boyer was fortunately introduced to the joy of writing theorem-provers in Lisp by his advisor, W. W. Bledsoe, who was a student of that practicing formalist A. P. Morse. Boyer's Ph.D. thesis was about locking, one of the many variants of resolution. This thesis was written while at the M.I.T. Artificial Intelligence Lab, during a year that Bledsoe spent a sabbatical there. Boyer next went, in 1971, to do research in the Metamathematics Unit at the University of Edinburgh, a Delphi and Mecca of automated reasoning and formal approaches to Artificial Intelligence. There Boyer began collaborating with Moore on their prover and many other projects, e.g., a text editor; their collaboration has continued at SRI International, the University of Texas at Austin, and Computational Logic, Inc., in Austin. Now they are working on recoding a new version of their prover in its own logic, a subset of applicative Common Lisp. Boyer has also done research at MCC in Austin. Currently, Boyer holds a professorship in computer sciences at the University of Texas at Austin, and a courtesy appointment in the math department there. Boyer is a member of the editorial boards of *Artificial Intelligence*, the *Journal of Automated Reasoning*, the *Journal of Logic and Computation*, Springer's new *Lecture Notes in Artificial Intelligence*, and Kluwer's new series on automated reasoning. He was general chairman of the 1984 conference on Lisp and Functional Programming. He is the co-author or editor of fewer than five books and 18 articles. His and Moore's prover, however, is available without fee by anonymous ftp from Internet site cli.com, and by running the prover on the examples supplied one can generate over twenty megabytes of proofs, with English commentary.