Chapter 1

Automated Reasoning and Bledsoe's Dream for the Field[1]

Larry Wos

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 60439

Abstract

In one sense, this article is a personal tribute to Woody Bledsoe. As such, the style will in general be that of private correspondence. However, since this article is also a compendium of experiments with an automated reasoning program, researchers interested in automated reasoning, mathematics, and logic will find pertinent material here. The results of those experiments strongly suggest that research frequently benefits greatly from the use of an automated reasoning program. As evidence, I select from those results some proofs that are better than one can find in the literature, and focus on some theorems that, until now, had never been proved with an automated reasoning program, theorems that Hilbert, Church, and various logicians thought significant. To add spice to the article, I present challenges for reasoning programs, including questions that are still open.

[1] This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

1.1 Coming Attractions

Woody, before getting deeply into the material, I feel certain that you would like to have a brief review of what is to come in this letter/article. I think I know you fairly well, and I think you know me fairly well. But, in the lattice of knowledge that one person has about another, greater than my knowledge of you and your knowledge of me is my knowledge of me—or, with regard to the focus of this article, of my research. Mathematics does indeed offer the use of some appropriate—or farfetched—metaphors.

I think of this document as a letter/article because it is partly a personal tribute to you—and hence, the style is often that of private correspondence—and partly a compendium of experiments with an automated reasoning program. At least to me, the results of those experiments strongly suggest that research frequently benefits greatly from the use of an automated reasoning program. Even the most skeptical should be rather startled at what can now be done with a reasoning program. Especially for you, I promise to select from those results some proofs that are better than one can find in the literature. I also promise to focus on some theorems that, until now, had never been proved with an automated reasoning program—theorems that Hilbert, Church, and various logicians thought worth their attention. Since—as you so well know—I also wish to interest others in what the field has to offer, some of my remarks and observations are directed to researchers in general.

The focus of this article is a significant fraction of your grand dream for automated reasoning. From your entrance into the field of automated reasoning until the present, you have daydreamed about a computer friend. Rather than the whole of your grand dream—which will take many years to realize, if ever—I shall concentrate on those aspects that are now a reality, reached far sooner than anyone could have guessed but two decades ago. Although these aspects—also present from the beginning in my dream for the field of automated reasoning—focus on but a fraction of your grander dream, nevertheless the fraction is itself a significant dream to have.

You envision an automated reasoning program that will, when instructed for the first time to prove one or more theorems, frequently succeed, and succeed without the researcher playing a key role. You dream of reasoning programs that contribute to mathematics or logic—by finding a better proof than expected, or producing a shorter proof than previously known, or answering an open question. And you hope to see mathematicians and logicians using reasoning programs as assistants in their research. In this article—although you will be hard to convince—I show that this significant fraction of your dream is in fact a reality. As evidence, I outline a fragment of the history of automated reasoning from its birth (approximately 1960) to the present, drawing mainly from experiences with automated reasoning programs designed by colleagues at Argonne National Laboratory. You and I both know that I shall focus mainly on recent results obtained with McCune's

powerful program OTTER [17]; I reserve my comments concerning other programs for Note 6. Even in your terms, I may succeed in staying with the truth—we shall see, yes?

Rather than a detailed account of people and places, the presentation consists of a sequence of snapshots showing how the effectiveness of automated reasoning gradually increased. The snapshots focus on individual questions, problems, and theorems, and the successes and failures resulting from their consideration by some automated reasoning program. By discussing some of the recent advances in theory, implementation, and application, I answer those who assert the impossibility of the effective automation of reasoning. How can such skeptics still exist?

To address the three cited objectives of your and my dream for an automated reasoning program and to illustrate how far automated reasoning has advanced, I give examples, examples of successfully proving theorems on the first try—and without much guidance from the user. A second set of examples focuses on finding better proofs than were expected, producing shorter proofs than were known, and answering open questions. The final examples concern the current use of an automated reasoning program by mathematicians and logicians.

The rate of recent advances suggests—certainly to me—that still more of your dream is within reach, as the field continues to offer intriguing challenges. Presented here are some of those challenges, including questions—still open—that are amenable to attack with an automated reasoning program. Although the progress is accelerating, there will always exist challenges for research in various areas of theory, implementation, and application.

Now at this point I can hear you saying, "Larry, I hear you; my students hear you; the world hears you. But, although I shouldn't be giving advice, I will. Tell me more, and tell the others who might be curious about the new results—expand on what you have just said, and don't mind a little repetition. And, oh yes, let me see in writing some of your opinions and biases, so we can all have a crack at them."

1.2 Motivation, Dedication, and Organization

As in any course, article, book, or meal, certain preliminaries are in order before the entree is served. However, if you prefer to feast immediately, you might simply fast-forward to the next section.

The choice of subject for this letter/article is based in part on many stimulating and enlightening conversations I have had with you, Woody, and is based in part on some recent successful experiments with McCune's automated reasoning program OTTER. By anticipating what you might say, let me now expand somewhat on the three cited goals that are part of your dream for automated reasoning. As will be shown with various examples, regarding

those important goals, your dream is now a reality.

First, you and I envision an automated reasoning program that will, when instructed for the first time to prove one or more theorems, frequently succeed, and succeed without the researcher playing a key role. Yes, I agree with the thought I am certain you have at this moment, the key point concerns the degree to which the success depends on the researcher's guidance. After all, you and I aim at a program that can function as a colleague—for example, finding a proof with little more than a hint from us. As evidence that this goal has been reached, I shall discuss (in Section 4) a successful attempt to use OTTER to prove 68 theorems suggested by Dana Scott, and (in Section 5) I shall include a new axiom system and a shorter proof. If you were focusing on your grand dream, you would at this point comment that we still do poorly with a randomly selected theorem from a typical advanced mathematics text, and you would be right. But think of it: 68 theorems in one run with OTTER, with no knowledge of any of the corresponding proofs and—of at least fair significance—theorems suggested by Scott.

You and I also dream of reasoning programs that contribute to mathematics or logic—by finding a better proof than expected, or producing a shorter proof than previously known, or—best of all—answering an open question. I shall give you examples of each of the three types of contribution. When you read the full story—which I include in this article—of how that already-mentioned new axiom system was actually found, I suspect you will consider the episode as evidence of contributing a bit to mathematics and logic. On the other hand, I suppose the story could also be used as evidence that at least one person interested in automated reasoning conducts research in an odd way. Some gambles win, and some lose—and, sometimes, it is hard to tell which has occurred.

And of course you and I hope to see mathematicians and logicians using reasoning programs as assistants in their research. Well, they now do, occasionally, use such programs, and some occurrences are briefly discussed.

Since I cannot with certainty rank the given three goals, I shall organize this article by simply considering them in the order cited. The promised outline of a fragment of the history of automated reasoning from its birth (approximately 1960) to the present will provide an appropriate perspective for a full appreciation of the significance of reaching the three objectives. Although from today's perspective the beginning was not particularly impressive or promising—I think you share this view—you may find that the evidence offered here shows that we have made impressive advances.

The examples have been chosen to reflect the state of the art at various points between 1960 and 1990, beginning with an example that today's researcher would correctly classify as offering no challenge, and ending with examples that—at least some—mathematicians and logicians consider interesting theorems. Far more interesting to me than history is the current state of the art, for it demonstrates that research does benefit greatly from the use

of an automated reasoning program. The examples will occasionally be accompanied by personal commentary concerning their significance. To answer those who assert the impossibility of the effective automation of reasoning, I also consider some of the recent advances in theory, implementation, and application.

For many of the examples, the corresponding input clauses are included and often even the clauses that illustrate a successful completion of the problem under study. To aid you in judging whether the dice were loaded or the deck stacked—even you might be a bit skeptical—I occasionally discuss the particular approach that was used to succeed, the parameter settings, and the reasons for choosing that approach. Again, occasional commentary is given. Finally, to stimulate research, I include problems that I consider challenging—or even surpassing—the limits of current automated reasoning programs, and offer questions—still open—that are amenable to attack with some existing program.

Throughout this article, in addition to proving theorems, I also discuss diverse ways in which a general-purpose, automated reasoning program can aid research; I focus mainly on McCune's marvelous and portable program OTTER. As you know, this program is based on the paradigm common to the programs designed by colleagues at Argonne. The lesser-known uses include the checking of given proofs, the discovery of problems for testing programs and ideas, the systematic search for shorter proofs, and the identification of errors in the chosen axiom system. Appropriate examples will be included here. For the more familiar use of proving theorems, examples will be given to show how a researcher can move from one attack to another in search of that which is sufficiently effective, thus illustrating the versatility offered by some automated reasoning programs.

For the final item of this section, I note that, throughout this article, I express various unsupported opinions, opinions based mostly on experimentation from 1964 to the present. I do so at your repeated encouragement. Indeed, when in conversation I have commented that I can give neither proof nor overwhelming data as the basis for certain utterances, you have replied that I need only issue an appropriate warning. So, Woody, here goes: This article contains commentary that is little more than opinion, and perhaps even bias. The commentary might add spice—for some, the taste of jalapeno peppers—to the presentation (in the following sections) of shorter proofs, new proofs, a new axiom system, new results, new techniques, and new uses of an automated reasoning program.

1.3 A Fragment of Automated Reasoning from Birth through Adolescence

This small taste of history provides a perspective for the following sections. In those sections I focus on the new material, and in them I give the hard evidence that three goals of your grander dream have been reached, at least to an important extent. The history presented in this section will also permit others to see how far away those goals were—even a few years ago—how far we have come, and (to some extent) how we got where we are.

The first problem (the Davis-Putnam example [2]) I identify with the field is, by today's standards, not very impressive. The Davis-Putnam example asks for a proof of the unsatisfiability of the following set of clauses. (For this and succeeding examples, the notation for clauses is that used to present a problem to OTTER, where "-" means **not** and "|" means **or**.)

```
P(x,y).
-P(y,f(x,y)) | -P(f(x,y),f(x,y)) | Q(x,y).
-P(y,f(x,y)) | -P(f(x,y),f(x,y)) | -Q(x,f(x,y) |
     -Q(f(x,y),f(x,y)).
```

Before the introduction of binary resolution [21], this problem was accepted as a challenge for reasoning programs. On the other hand, with the use of binary resolution, a proof was obtainable in 1963 in essentially no CPU time. Of course, rather than deprecating the quality of the early research, the simplicity of the Davis-Putnam example illustrates what little power was offered by the programs existing before 1963. The problem did serve nicely as a beginning, and—more important to me—the ease with which it was solved with a reasoning program motivated my attempt to obtain computer proofs of theorems from the mathematics literature.

The first theorem that was tried is a simple classroom exercise: If in a group the square of every element x is the identity e, the group is commutative. The attempt to prove the theorem failed; after 2,000 clauses were retained, memory was exhausted (on an IBM 704 computer). Since the failure led to the formulation of the set of support strategy [31], I am still fond of the theorem, even though I recognize its utter simplicity. Since the importance of strategy in general and of the set of support strategy in particular has been repeatedly proven [34], I am continually disappointed at the lack of research directed to that aspect of automated reasoning. To stimulate such research—which was, I admit, one reason I wrote my second book [36]—you and I could pool our personal resources and offer a $20 prize for a significant contribution in the area of strategy.

The first attempt to prove that, in a ring, the product of $-x$ and $-y$ is xy also failed. The failure led to the use of lemma adjunction, specifically, the lemmas that assert that the product in either order of 0 and x is 0.

The first attempt to prove that subgroups of index 2 are normal failed. To succeed, a primitive form of case analysis was used; one case focused on an element in the subgroup, the other case on an element outside the subgroup. I find this theorem appealing, for it offers some difficulty for a person to prove, and it is somewhat significant from the viewpoint of mathematics. The index 2 problem has added appeal for me, for the attack that succeeded is representative of the way I prefer case analysis to be treated, rather than, say, by automatically splitting every ground clause into its individual literals.

Even at the end of the sixties—because of the lack of power offered by the then-current reasoning programs—I knew it was futile to attempt to seek a computer proof of the "commutator theorem": If in a group the cube of every x is the identity e, then $[[x,y],y] = e$ for all x and y, where the commutator $[x,y]$ of any two elements x and y of a group is the product of x, y, the inverse of x, and the inverse of y. Indeed, although I presented this theorem as an example of the potential use of paramodulation [20], its included clause-notation proof was obtained by hand. As it turned out, the first computer proof of the theorem was obtained with hyperresolution [22]; the first paramodulation proof by computer was not obtained until approximately six years after the problem was suggested. This theorem is now, and will always be, important to me, for its study solidified my interest in paramodulation [33], an interest that has never diminished. The theorem also holds special meaning for me, for Ross Overbeek's reading of the corresponding paper prompted him to enter the field; any estimate of the value of that occurrence falls short of the mark.

The four theorems and the results were typical of the 1960s, at least of the research I shared with colleagues. Failure was the expected outcome, and we were seldom disappointed. In other words, the first of your goals for the field of automated reasoning—succeeding moderately often on the first attempt to prove a theorem not tried before—was far in the future, if reachable at all. As it turned out, and as I shall exemplify in the next section, two decades after the close of the 1960s would see the attainment of this goal. Examples of success on the first attempt include theorems of interest to Hilbert, Church, Frege, Bernays, Lukasiewicz, and Tarski.

The second goal—a reasoning program making contributions to mathematics and logic—brings me even more excitement than does the first goal. However, except for a brief consideration in the late 1960s of attempting to answer an open question concerning the independence of the first three axioms of the five for a ternary Boolean algebra [30], I was not involved in such an activity until the late 1970s. In fact, except for the discovery by Guard and associates of SAM's lemma [4], I was almost certain such contributions would not occur in my lifetime. Clearly—and most fortunately—I was in error, for the period from 1978 to the present witnessed the answering of open questions from various fields of mathematics and logic, where the answers were obtained with substantial assistance from an automated reasoning program. The fields include combinatory logic, finite semigroups, ternary Boolean alge-

bra, Robbins algebra, equivalential calculus, and various areas of logic related to implicational calculus.

As for your third goal—the actual use by mathematicians and logicians of automated reasoning programs—until the early 1980s, the only item that occurs to me concerns Paul Halmos and his visit to the University of Texas. On that visit—and I checked this with Halmos—you and Halmos, using a theorem prover, were able to prove a theorem from *Naive Set Theory* [5]. Unfortunately, my best recollection says that the theorem occurs between pages 43 and 45; perhaps you can supply the needed details. I am not sure this early 1970s incident actually qualifies for "use of an automated reasoning program by a mathematician", but it is in the vicinity.

On the other hand, without doubt, Kalman's use at Argonne of the program ITP [12, 13] during his stay in the early 1980s clearly qualifies; Kalman was studying various aspects of equivalential calculus, in part in the context of group theory. Kalman continues to conduct research with the aid of a reasoning program; he is now using OTTER for additional studies in group theory. For a second example, Scott has just successfully used OTTER (on his Macintosh) to obtain a proof of the completeness of a Lukasiewicz axiom system for sentential calculus.

History is treating our goals well—or do you remain understandably skeptical? Resist if you can, but I suggest such skepticism may be difficult to maintain when you see what happened here in August 1990, research prompted by Scott's stimulating communication by email. Of course, there will always exist research problems to solve in various areas of automated reasoning. We would not wish it any other way, for we have each spent a major fraction of our lives in this field. We have certainly shown that the automation of careful reasoning is far more than feasible, contrary to those who assert its impossibility. Although the type of reasoning found in mathematics is indeed deep and beyond complete capturing, we have come much closer than I would have guessed when I entered the field in 1963.

Clearly—and this remark is particularly appropriate when addressed to me—care must be exercised to control the obvious enthusiasm; the last time I exercised such care occurred in approximately 1802. Yes, I am aware that your grand dream is far from realized and that many graduate students could outperform the best of the existing automated reasoning programs if the competition centered on randomly selected theorems from mathematics texts. But how many of those students have answered an open question—even further, a question posed by Kaplansky? As you well know, such a question—as well as other open questions—has been answered with crucial assistance from one of Argonne's reasoning programs.

Exercising my usual conservative approach, which brings to eye, ear, and mind a herd of stampeding elephants—just between you and me—the results of the next two sections are truly exciting and beautiful!

1.4 Successes on the First Try: Goal 1

In this and succeeding sections—finally getting to new results and new uses—I am going to emphasize the most recent experiences I have had with OTTER, experiences involving theorems suggested by Dana Scott. The theorems are from various logical calculi, such as sentential calculus. Beginning in 1879, such theorems have occupied the attention of Frege, Russell, Hilbert, Tarski, Bernays, Lukasiewicz, Church, and others of their stature; two books that contain pertinent material are [10, 11]. I shall try hard to tell the truth about how the successes occurred, under what conditions, in what order, and how easily for my colleague McCune and me. However, when I am experimenting and not in the presence of one of my colleagues, I can easily lose the fine details. But, you need not worry; most, if not all, of what I present can be independently verified, so the loss of some of the minute history is not devastating.

When Scott returned home after a visit to Argonne in August 1990, he sent by email 68 theorems for OTTER's attempt at proof. The theorems (whose negations are given shortly) are numbered 4 through 71 in a study by Lukasiewicz; they are provable from the following three axioms expressed in clause notation. The first of the three can be thought of as transitivity, $(x \rightarrow y) \rightarrow ((y \rightarrow z) \rightarrow (x \rightarrow z))$. The second can be read as $(\neg x \rightarrow x) \rightarrow x$, and the third as $x \rightarrow (\neg x \rightarrow y)$.

```
(L1)  P(i(i(x,y),i(i(y,z),i(x,z)))).
(L2)  P(i(i(n(x),x),x)).
(L3)  P(i(x,i(n(x),y))).
```

The rule of inference is condensed detachment, captured with the following clause when hyperresolution is the chosen inference rule.

```
-P(i(x,y)) | -P(x) | P(y).
```

Lukasiewicz accurately claims that the given three axioms are complete for sentential calculus. An axiom set is complete for sentential calculus if the set of formulas that can be deduced from it with the use of condensed detachment and instantiation is precisely the set of formulas (in effect, unit clauses) that are true under all assignments of true and false, with *implication* and *negation* used as logicians use these terms. Therefore, the following 68 theorems, presented as negations to seek a contradiction, must hold. Frequently, throughout the remainder of this article, the theorems in the unnegated form will be referred to as theses. Frequently also, the ANSWER literal (which is ignored by the program when applying an inference rule) is appended to a clause to indicate the role of the clause. The ANSWER literal will usually be abbreviated with "$ANS", the word "thesis" with "th", the word "negated" with "neg", and the word "binary" with "bin".

```
-P(i(i(i(i(q,r),i(p,r)),s),i(i(p,q),s))) | $ANS(neg_th_04).
-P(i(i(p,i(q,r)),i(i(s,q),i(p,i(s,r))))) | $ANS(neg_th_05).
-P(i(i(p,q),i(i(i(p,r),s),i(i(q,r),s)))) | $ANS(neg_th_06).
-P(i(i(t,i(i(p,r),s)),i(i(p,q),i(t,i(i(q,r),s)))))
  | $ANS(neg_th_07).
-P(i(i(q,r),i(i(p,q),i(i(r,s),i(p,s))))) | $ANS(neg_th_08).
-P(i(i(i(n(p),q),r),i(p,r))) | $ANS(neg_th_09).
-P(i(p,i(i(i(n(p),p),p),i(i(q,p),p)))) | $ANS(neg_th_10).
-P(i(i(q,i(i(n(p),p),p)),i(i(n(p),p),p))) | $ANS(neg_th_11).
-P(i(t,i(i(n(p),p),p))) | $ANS(neg_th_12).
-P(i(i(n(p),q),i(t,i(i(q,p),p)))) | $ANS(neg_th_13).
-P(i(i(i(t,i(i(q,p),p)),r),i(i(n(p),q),r))) | $ANS(neg_th_14).
-P(i(i(n(p),q),i(i(q,p),p))) | $ANS(neg_th_15).
-P(i(p,p)) | $ANS(neg_th_16).
-P(i(p,i(i(q,p),p))) | $ANS(neg_th_17).
-P(i(q,i(p,q))) | $ANS(neg_th_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANS(neg_th_19).
-P(i(p,i(i(p,q),q))) | $ANS(neg_th_20).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(neg_th_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(neg_th_22).
-P(i(i(i(q,i(p,r)),s),i(i(p,i(q,r)),s))) | $ANS(neg_th_23).
-P(i(i(i(p,q),p),p)) | $ANS(neg_th_24).
-P(i(i(i(p,r),s),i(i(p,q),i(i(q,r),s)))) | $ANS(neg_th_25).
-P(i(i(i(p,q),r),i(i(r,p),p))) | $ANS(neg_th_26).
-P(i(i(i(p,q),q),i(i(q,p),p))) | $ANS(neg_th_27).
-P(i(i(i(i(r,p),p),s),i(i(i(p,q),r),s))) | $ANS(neg_th_28).
-P(i(i(i(p,q),r),i(i(p,r),r))) | $ANS(neg_th_29).
-P(i(i(p,i(p,q)),i(p,q))) | $ANS(neg_th_30).
-P(i(i(p,s),i(i(i(p,q),r),i(i(s,r),r)))) | $ANS(neg_th_31).
-P(i(i(i(p,q),r),i(i(p,s),i(i(s,r),r)))) | $ANS(neg_th_32).
-P(i(i(p,s),i(i(s,i(q,i(p,r))),i(q,i(p,r))))) | $ANS(neg_th_33).
-P(i(i(s,i(q,i(p,r))),i(i(p,s),i(q,i(p,r))))) | $ANS(neg_th_34).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(neg_th_35).
-P(i(n(p),i(p,q))) | $ANS(neg_th_36).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANS(neg_th_37).
-P(i(i(p,n(p)),n(p))) | $ANS(neg_th_38).
-P(i(n(n(p)),p)) | $ANS(neg_th_39).
-P(i(p,n(n(p)))) | $ANS(neg_th_40).
-P(i(i(p,q),i(n(n(p)),q))) | $ANS(neg_th_41).
-P(i(i(i(n(n(p)),q),r),i(i(p,q),r))) | $ANS(neg_th_42).
-P(i(i(p,q),i(i(q,n(p)),n(p)))) | $ANS(neg_th_43).
-P(i(i(s,i(q,n(p))),i(i(p,q),i(s,n(p))))) | $ANS(neg_th_44).
-P(i(i(s,i(q,p)),i(i(n(p),q),i(s,p)))) | $ANS(neg_th_45).
-P(i(i(p,q),i(n(q),n(p)))) | $ANS(neg_th_46).
```

```
-P(i(i(p,n(q)),i(q,n(p)))) | $ANS(neg_th_47).
-P(i(i(n(p),q),i(n(q),p))) | $ANS(neg_th_48).
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(neg_th_49).
-P(i(i(i(n(q),p),r),i(i(n(p),q),r))) | $ANS(neg_th_50).
-P(i(i(p,i(q,r)),i(p,i(n(r),n(q))))) | $ANS(neg_th_51).
-P(i(i(p,i(q,n(r))),i(p,i(r,n(q))))) | $ANS(neg_th_52).
-P(i(i(n(p),q),i(i(p,q),q))) | $ANS(neg_th_53).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANS(neg_th_54).
-P(i(i(p,q),i(i(p,n(q)),n(p)))) | $ANS(neg_th_55).
-P(i(i(i(i(p,q),q),r),i(i(n(p),q),r))) | $ANS(neg_th_56).
-P(i(i(n(p),r),i(i(p,q),i(i(q,r),r)))) | $ANS(neg_th_57).
-P(i(i(i(i(p,q),i(i(q,r),r)),s),i(i(n(p),r),s))) |
      $ANS(neg_th_58).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(neg_th_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) |
      $ANS(neg_th_60).
-P(i(i(p,r),i(i(q,r),i(i(n(p),q),r)))) | $ANS(neg_th_61).
-P(i(i(n(n(p)),q),i(p,q))) | $ANS(neg_th_62).
-P(i(q,i(p,p))) | $ANS(neg_th_63).
-P(i(n(i(p,p)),q)) | $ANS(neg_th_64).
-P(i(i(n(q),n(i(p,p))),q)) | $ANS(neg_th_65).
-P(i(n(i(p,q)),p)) | $ANS(neg_th_66).
-P(i(n(i(p,q)),n(q))) | $ANS(neg_th_67).
-P(i(n(i(p,n(q))),q)) | $ANS(neg_th_68).
-P(i(p,i(n(q),n(i(p,q))))) | $ANS(neg_th_69).
-P(i(p,i(q,n(i(p,n(q)))))) | $ANS(neg_th_70).
-P(n(i(i(p,p),n(i(q,q))))) | $ANS(neg_th_71).
```

My esteemed colleague McCune, with whom I have conducted most of the experiments reported here, immediately submitted the 68 theorems to OTTER. He chose hyperresolution as the only inference rule; he placed the clause for condensed detachment in the axioms, the three Lukasiewicz axioms in the set of support, and the 68 negations in the passive list—a list that is inactive except for both testing for the completion of a proof by detecting unit conflict and checking for forward subsumption [21]. For the weighting strategy [15, 34, 36]—the strategy used by OTTER to decide where next to focus its attention—he chose to use symbol count. The motivation was to approach the theorems as simply as possible, keying on the three Lukasiewicz axioms, using the 68 negations only to detect the corresponding proofs (if found), and causing OTTER's search for proofs to be directed by focusing at any given point on the shortest formula available. OTTER proved 33 of the theorems before it was decided that more CPU time would yield little additional. A similar attempt with ROO, a parallel version of OTTER, produced 48 proofs.

Motivated by the strong urge to demonstrate OTTER's power by sending proofs of all 68 theorems at once to Scott, I decided to make one additional

attempt. The attempt was based on replacing the use of symbol count (to direct OTTER in its choice of where next to focus its attention) by a set of 68 weight templates, each matching the corresponding pattern of occurrences of the function i in one of the theorems under attack. I reasoned—hazarded is probably more accurate—that, if the 68 theorems were the steps in a proof of a desired result, then each merited emphasis if and when deduced.

It worked, and better than I would have guessed; OTTER proved all 68 theorems in less than 16 CPU minutes on a SPARCstation. As it turned out, Scott was interested mainly in eight of the theses: 16, 18, 24, 21, 35, 39, 40, and 49. He wished to have a single proof of all eight, rather than piecing eight separate results together and tediously removing duplicate steps and such, which he could compare to the Lukasiewicz proof for style differences. A ploy that succeeded focuses on taking the disjunction of the corresponding eight negations and placing it among the axioms; the desired proof is found when the empty clause is generated with the disjunction playing the role of nucleus. He informed me that the resulting 46-step proof more or less matches the style of that given by Lukasiewicz.

But the story does not end there. Its ending is given in the next section, where it properly belongs, as an example of a small contribution to logic made by an automated reasoning program.

Although three attempts were required to obtain all 68 proofs, the study led to the formulation of an approach—not scientifically justified—that was used on the following example. The theorem in question asks for a proof that the third axiom (FL3) of the Frege/Lukasiewicz axiom system FL (presented by Church) follows from the six axioms (F1) through (F6), Frege's original axiom system F for sentential calculus. (The first two of the three axioms—theses 18, 35, and 49—of which FL consists are, respectively, the first and second of F.) When OTTER considered the following seven clauses, coupled with the clause for condensed detachment, it found the desired proof on the first try.

```
(F1 th_18)  P(i(x,i(y,x))).
(F2 th_35)  P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(F3 th_21)  P(i(i(x,i(y,z)),i(y,i(x,z)))).
(F4 th_39)  P(i(i(x,y),i(n(y),n(x)))).
(F5 th_40)  P(i(n(n(x)),x)).
(F6 th_46)  P(i(x,n(n(x)))).
(FL3 neg_th_49)  -P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_FL3).
```

The approach was again to rely on the weight templates whose pattern of the function i matches the 68 theses just discussed.

At this point, you might properly wonder what possible explanation I can give for such an action, especially since I was seeking but one of the 68 proofs, that of 49. Explanation: if the use of those 68 patterns as weight templates enabled McCune and me to send to Scott the proofs of the test problems

he suggested, then those patterns must be good ones, good ones for other theorems in this area of logic—even if the axioms (hypotheses) were changed. Reasoning of this type accounts for the fact that some gamblers win great sums; it also accounts for the fact that most gamblers lose their entire stake. As it turned out, the sought-after proof was obtained in less than 1 CPU second, a proof of 14 steps. Since the proof was obtained on the first attempt, do we not have overwhelming evidence of the realization of the first goal, or is the evidence—at least to this point—underwhelming?

I then took for weight templates the patterns suggested by the 14 steps of the proof, set the appropriate flags to instruct OTTER to seek possibly shorter proofs of (FL3), and obtained in one run the following 11-step proof in 6 CPU seconds. (In Note 5, I have more to say concerning the importance of seeking shorter proofs and the use of weight templates.) The number in the first field gives the corresponding position of the clause among those retained during OTTER's attempt. The expression *hyper,x,y,z* typically denotes that x is the clause in focus, y is the nucleus, z is the other satellite (for condensed detachment), and hyperresolution is the inference rule.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] P(i(x,i(y,x))).
3 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
4 [] P(i(i(x,i(y,z)),i(y,i(x,z)))).
5 [] P(i(i(x,y),i(n(y),n(x)))).
6 [] P(i(n(n(x)),x)).
7 [] P(i(x,n(n(x)))).
8 [] -P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_FL3).
--------------------
22 [hyper,4,1,3] P(i(i(x,y),i(i(x,i(y,z)),i(x,z)))).
25 [hyper,5,1,4] P(i(n(x),i(i(y,x),n(y)))).
33 [hyper,6,1,2] P(i(x,i(n(n(y)),y))).
50 [hyper,22,1,7] P(i(i(x,i(n(n(x)),y)),i(x,y))).
56 [hyper,25,1,2] P(i(x,i(n(y),i(i(z,y),n(z))))).
61 [hyper,33,1,3] P(i(i(x,n(n(y))),i(x,y))).
86 [hyper,56,1,50] P(i(x,i(i(y,n(x)),n(y)))).
523 [hyper,61,1,2] P(i(x,i(i(y,n(n(z))),i(y,z)))).
541 [hyper,523,1,3] P(i(i(x,i(y,n(n(z)))),i(x,i(y,z)))).
675 [hyper,86,1,4] P(i(i(x,n(y)),i(y,n(x)))).
708 [hyper,675,1,541] P(i(i(n(x),n(y)),i(y,x))).
```

Clause (708) contradicts clause (8), and the proof is complete.

I have no idea whether this proof is the same as, equivalent to, or better than that to be found in the literature.

The motivation for studying the next theorem—another example of succeeding on the first try—was simple curiosity on my part. If Scott was satisfied that OTTER proved the completeness of the three-axiom system of

Lukasiewicz by proving the eight theses, 16, 18, 24, 21, 35, 39, 40, and 49, then an obvious theorem to attempt to prove is the converse: from these eight axioms, one can deduce each of (L1), (L2), and (L3). (As we later learned, for a complete system, one need consider only the second, fifth, and eighth; in other words, the other five are dependent on these three.) The same approach was used again, directing OTTER's search for the desired three proofs by using the 68 weight templates. The theses were proved in a single run, in the order (L3), (L1), and (L2). The proofs were obtained in the interval of CPU seconds between 97 and 101.

You might find the following example of particular interest, for it dramatically demonstrates how far we have come and exhibits OTTER's nearly linear performance with respect to clause generation. The theorem to prove asserts that the third of the following six axioms, Frege's system for sentential calculus, is dependent on the remaining five.

```
(F1 th_18)  P(i(x,i(y,x))).
(F2 th_35)  P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(F3 th_21)  P(i(i(x,i(y,z)),i(y,i(x,z)))).
(F4 th_39)  P(i(i(x,y),i(n(y),n(x)))).
(F5 th_40)  P(i(n(n(x)),x)).
(F6 th_46)  P(i(x,n(n(x)))).
```

Using the approach described in this section, but with a smaller set of weight templates, OTTER succeeded in proving—on the first attempt—that the third axiom is indeed dependent on the other five.

For the following reasons, this first-attempt success is (to me) both stimulating and enlightening. The proof was obtained (on a SPARCstation) in just under 19 CPU hours. The proof was completed upon retention of a clause numbered 11331, after generating over 30,000,000 clauses. The performance of OTTER was nearly linear; during the first few CPU seconds, clauses were generated at the rate of 550 per CPU second, and during the last few CPU seconds, clauses were generated at the rate of 460 per CPU second. I could not resist letting OTTER run for additional time to see what would happen to the generation rate. At the 43 CPU hour mark, after generating just under 66,000,000 clauses, the rate had finally dropped to 300 clauses per CPU second—pretty impressive to see such small degradation over such a long period of time. A fast, tireless, and *accurate* assistant, OTTER provides a nice complement to the researcher with insight. The mathematician or logician might make marvelous discoveries, after mastering and then using this program. The proof OTTER found consists of 74 steps, not counting, of course, the five (input) axioms, all of which are used.

I have a tag or postscript to the discussion of this Frege dependence theorem. As OTTER was searching for a proof that the third axiom is dependent on the other five, my colleague McCune informed me (after browsing in one of Lukasiewicz's books) that the dependence could be proved from the first

two axioms alone. On a second computer, OTTER was asked to search for such a proof. We were motivated, of course, by curiosity about which run would finish first—the long run just discussed, or this second attempt—and by the wish to compare the two proofs, should they be found. The second attempt succeeded first, in under 52 CPU seconds, finding a proof consisting of 11 steps. Yes, for this newer theorem, one attempt was again sufficient. Considered together, the two results suggest that the presence of unneeded but usable axioms can sharply detract from a program's effectiveness.

The final detailed example of success on the first try concerns proving the completeness of the axiom system consisting of (the unnegated form of) theses 19, 37, and 59. McCune informed me of this system from his reading of Lukasiewicz. Scott has commented to me that the system is indeed interesting—and he can explain why—but that Lukasiewicz does not explain why nor does he give the corresponding proofs. OTTER's proof that theses 19, 37, and 59 form a complete system rests on the deduction of (FL1), (FL2), and (FL3), mentioned earlier as a complete system. Since these proofs may not exist in the literature, OTTER's proofs are included here. The following three proofs are, respectively, for the deduction of (FL1), (FL2), and (FL3); the third required approximately 406 CPU seconds, and the first two required less than 1 CPU second.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] P(i(i(i(x,y),z),i(y,z))).
21 [] -P(i(q,i(p,q))) | $ANS(step_th_18).
--------------------
35 [hyper,2,1,2] P(i(x,i(y,x))).
```

Clause (35) contradicts clause (21), and the proof is complete.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] P(i(i(i(x,y),z),i(y,z))).
3 [] P(i(i(i(x,y),z),i(n(x),z))).
4 [] P(i(i(n(x),z),i(i(y,z),i(i(x,y),z)))).
24 [] -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(step_th_35).
--------------------
35 [hyper,2,1,2] P(i(x,i(y,x))).
43 [hyper,35,1,3] P(i(n(x),i(y,i(x,z)))).
44 [hyper,35,1,2] P(i(x,i(y,i(z,x)))).
68 [hyper,43,1,4] P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))).
72 [hyper,68,1,35] P(i(i(x,i(x,y)),i(z,i(x,y)))).
79 [hyper,72,1,43] P(i(x,i(n(y),i(y,z)))).
81 [hyper,79,1,79] P(i(n(x),i(x,y))).
87 [hyper,81,1,4] P(i(i(x,i(y,z)),i(i(y,x),i(y,z)))).
98 [hyper,87,1,44] P(i(i(x,y),i(x,i(z,y)))).
299 [hyper,98,1,68] P(i(i(x,i(y,z)),i(y,i(x,z)))).
```

```
338 [hyper,299,1,98] P(i(x,i(i(x,y),i(z,y)))).
1127 [hyper,338,1,68] P(i(i(x,y),i(i(y,z),i(x,z)))).
1208 [hyper,1127,1,299] P(i(i(x,y),i(i(z,x),i(z,y)))).
1285 [hyper,1208,1,68] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
```

Clause (1285) contradicts clause (24), and the proof is complete.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] P(i(i(i(x,y),z),i(y,z))).
3 [] P(i(i(i(x,y),z),i(n(x),z))).
4 [] P(i(i(n(x),z),i(i(y,z),i(i(x,y),z)))).
27 [] -P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_th_49).
--------------------
35 [hyper,2,1,2] P(i(x,i(y,x))).
42 [hyper,35,1,4] P(i(i(x,i(y,n(z))),i(i(z,x),i(y,n(z))))).
43 [hyper,35,1,3] P(i(n(x),i(y,i(x,z)))).
44 [hyper,35,1,2] P(i(x,i(y,i(z,x)))).
48 [hyper,42,1,35] P(i(i(x,n(x)),i(y,n(x)))).
68 [hyper,43,1,4] P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))).
72 [hyper,68,1,35] P(i(i(x,i(x,y)),i(z,i(x,y)))).
79 [hyper,72,1,43] P(i(x,i(n(y),i(y,z)))).
81 [hyper,79,1,79] P(i(n(x),i(x,y))).
87 [hyper,81,1,4] P(i(i(x,i(y,z)),i(i(y,x),i(y,z)))).
98 [hyper,87,1,44] P(i(i(x,y),i(x,i(z,y)))).
101 [hyper,87,1,35] P(i(i(x,y),i(x,y))).
104 [hyper,101,1,87] P(i(i(x,i(x,y)),i(x,y))).
111 [hyper,104,1,48] P(i(i(x,n(x)),n(x))).
112 [hyper,104,1,35] P(i(x,x)).
299 [hyper,98,1,68] P(i(i(x,i(y,z)),i(y,i(x,z)))).
338 [hyper,299,1,98] P(i(x,i(i(x,y),i(z,y)))).
1127 [hyper,338,1,68] P(i(i(x,y),i(i(y,z),i(x,z)))).
1203 [hyper,1127,1,1127] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
1208 [hyper,1127,1,299] P(i(i(x,y),i(i(z,x),i(z,y)))).
1234 [hyper,1127,1,299] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
1257 [hyper,1203,1,1203] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
1260 [hyper,1203,1,1127] P(i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))).
1321 [hyper,1208,1,4] P(i(i(x,i(n(y),z)),
                        i(x,i(i(u,z),i(i(y,u),z))))).
1492 [hyper,1257,1,1234] P(i(i(x,i(y,z)),i(i(u,x),i(y,i(u,z))))).
1517 [hyper,1260,1,299] P(i(i(i(x,y),z),i(i(x,u),i(i(u,y),z)))).
1640 [hyper,1321,1,3] P(i(i(i(x,y),z),i(i(u,z),i(i(x,u),z)))).
2266 [hyper,1640,1,1492] P(i(i(x,i(i(y,z),u)),
                            i(i(v,u), i(x,i(i(y,v),u))))).
2269 [hyper,1640,1,299] P(i(i(x,y),i(i(i(z,u),y),i(i(z,x),y)))).
2751 [hyper,2269,1,112] P(i(i(i(x,y),z),i(i(x,z),z))).
```

```
3002 [hyper,2751,1,1517] P(i(i(i(x,y),z),i(i(z,u),i(i(x,u),u)))).
3204 [hyper,3002,1,111] P(i(i(n(x),y),i(i(x,y),y))).
3331 [hyper,3204,1,299] P(i(i(x,y),i(i(n(x),y),y))).
3544 [hyper,3331,1,2266] P(i(i(x,y),i(i(z,y),i(i(n(z),x),y)))).
3879 [hyper,3544,1,299] P(i(i(x,y),i(i(z,y),i(i(n(x),z),y)))).
3982 [hyper,3879,1,35] P(i(i(x,i(y,z)),i(i(n(z),x),i(y,z)))).
4126 [hyper,3982,1,81] P(i(i(n(x),n(y)),i(y,x))).
```

Clause (4126) contradicts clause (27), and the proof is complete.

In 1923, Hilbert offered as a complete axiom system for sentential calculus that consisting of theses 18, 21, 22, 30, 3, and 54. Later, it was proved that 30 is dependent on the remaining axioms in Hilbert's system. On the first try, OTTER proved (with a 3-step proof) the dependence of 30 in less than 2 CPU seconds and, in that same run, proved the completeness of Hilbert's system. The first completeness proof proceeds by deducing the system consisting of theses 19, 37, and 59, completing a 9-step proof in just over 7 CPU seconds. In just under 17 CPU seconds, OTTER found a different 28-step proof of completeness by deducing the system *FL* consisting of theses 18, 35, and 49. However, later in the same run, a far shorter proof was found in approximately 1,733 CPU seconds, a proof consisting of 12 steps. Still in the same run, a 30-step proof of the completeness of Hilbert's system was found by deducing Frege's system *F* (excluding the dependent axiom) in approximately 143 CPU seconds. A shorter 16-step proof was then found in approximately 3,706 CPU seconds. A 6-step proof of the Lukasiewicz system was obtained in approximately 961 CPU seconds, and a 5-step proof in approximately 1,848 CPU seconds.

A few observations concerning these results are now in order. The way OTTER can be used is strikingly different from the approach typically taken with a colleague; Notes 4 and 7 elaborate on this point. In the latter case, a decision is made concerning which single path is best to pursue; for example, if the completeness of the Hilbert axiom system is the objective, the decision might be to attempt to prove the Lukasiewicz system or, instead, to attempt to prove Frege's. In contrast, with OTTER as your assistant, a good approach consists of asking this program to consider simultaneously a number of possible paths to a completeness proof; more comments are given in Note 5. Indeed, as evidenced in the preceding paragraph, that approach was markedly successful—one run produced a number of different completeness proofs for the axiom set under study. Among those many proofs, some differ in the intermediate steps but are alike in that each completes by deducing all the members of a given complete system. In contrast, other proofs obtained in that single run complete by deducing the members of a totally different complete system.

To me it seems clear that, at least in some areas, all has changed from the early 1960s, from the mid-1970s, and even from five years ago. Indeed, as the following summary suggests—and it may not convince you, for you

can drive a hard bargain—certain important aspects of your dream for the field are now a reality. An unsophisticated approach in which one chooses the obvious inference rule and obvious weights to direct the search (by symbol count) yields many proofs on the first try. With a little bit of sophistication, an approach can be formulated, based on the study of one set of theorems, and then used to obtain on the first try proofs of theorems that merited the attention of great minds. You and I know that in no way do the results of the cited experiments lessen the achievements of the various logicians involved in the studies beginning with Frege in 1879. Instead—at least I would like to find that this is the case—the successes with OTTER mark a singular achievement for automated reasoning and suggest that the field has indeed advanced significantly.

1.5 Contributions to Mathematics and Logic: Goal 2

Since this article is in part written for you, to add a special touch, I present new material and, further, material obtained during the writing of this article. The new results include an axiom system that may not have been seen before, and what appears to be a proof far shorter than is found in the literature; each was obtained with the aid of the program OTTER. To complement the new material and to give additional evidence of the possible value of automated reasoning to mathematics and logic, I shall begin by briefly reviewing some open questions that were answered with the assistance of a reasoning program. Some of these questions might prove challenging and useful for those who wish to compare the effectiveness of various programs, to evaluate the potential of a new approach, or—by attempting to answer the questions unaided by a computer friend—to gain an appreciation for how far the field has come.

Of course, I can now hear you asking one of your provocative questions. "Larry, before you give your review of the past ten or so years, how far *has* the field come?" The accurate answer to your question, it seems to me, depends on whether the focus is on how far we *have* traveled or on how far we have *to* travel. To graphically see what I mean, we can look to the space program. In particular, the man in the moon is no longer a myth; indeed a person has walked on it. But the distance to the moon is dwarfed by the distance to the outermost planet, which is in turn miniscule when compared to the distance to another galaxy. So, how far has the space program advanced? Four decades ago, I would have doubted that we would reach the moon; two decades ago, I did doubt that an automated reasoning program would play a key role in answering open questions. Those two misplaced doubts have not shaken me; I still have doubts. For one example, if Texas is larger than Illinois—and, mind you, I said if—it is to make room for taller people, for tall Texans.

As you know, beginning in 1978, automated reasoning programs designed

by members of the Argonne group have been used to answer various open questions, in finite semigroups posed by Kaplansky [29], in equivalential calculus posed by Kalman [35], in ternary Boolean algebra posed by Grau [30], in finite semigroups posed by McFadden [14], and in combinatory logic posed by Smullyan [24, 25, 37]. In my opinion, the successes in combinatory logic provide the most powerful evidence that automated reasoning has made contributions of note.

Barendregt [1] defines combinatory logic as an equational system satisfying the combinators S and K, where the respective actions of the constants S and K are given by the following two clauses.

```
EQUAL(a(a(a(S,x),y),z),a(a(x,z),a(y,z))).
EQUAL(a(a(K,x),y),x).
```

The Smullyan questions (answered by my colleague McCune and me) focus on what are called *fragments* of combinatory logic, and not the logic in its entirety. One of Smullyan's questions concerns finding, if such exists, an appropriate combinator to be used with the combinator B (defined with the following clause) to construct a fixed point combinator.

```
EQUAL(a(a(a(B,x),y),z),a(x,a(y,z))).
```

The combinator Θ is a *fixed point combinator* if and only if, for all combinators x, the equation $\Theta x = x(\Theta x)$ holds. The *strong fixed point property* holds for the fragment with basis **B** if and only if there exists a fixed point combinator Θ such that Θ is expressed purely in terms of the elements of B.

Statman answered Smullyan's question in the affirmative [26] by using the combinator W, defined by the following clause.

```
EQUAL(a(a(W,x),y),a(a(x,y),y)).
```

In addition to Statman's combinator, the fourth in the following list of five, McCune and I found four others by using an automated reasoning program [16].

$$\Theta_1 = B(B(B(WW)W)B)B$$
$$\Theta_2 = B(B(WW)W)(BBB)$$
$$\Theta_3 = B(B(WW)(BWB))B$$
$$\Theta_4 = B(WW)(BW(BBB))$$
$$\Theta_5 = B(WW)(B(BWB)B)$$

Smullyan did indeed seem impressed by the discovery of the additional four. Later, by using the *kernel strategy* [37] applied by OTTER, we found an infinite class of infinite sets of such combinators. That result is still extremely gratifying to me, especially in my role as mathematician. Indeed, to find such richness where, just one year earlier, it was not known whether any appropriate combinators existed, and then to make the discovery because of

formulating a new strategy that could be effectively applied by an automated reasoning program, signaled to me a significant advance in the field.

If you are curious and immediately wonder whether we simply built on Statman's combinator, I can accurately report that we did not; indeed, we found other combinators to act in the place of W. In particular, recalling that expressions in combinatory logic are assumed to be left associated unless otherwise indicated, our combinator N and the combinator H serve well.

$$Nxyz = xzyz$$
$$Hxyz = xyzy$$

The proofs—obtained with OTTER applying the kernel strategy—that the fragments with respective bases of combinators B and N and of B and H satisfy the strong fixed point property were completed by finding the following [37].

$$B(B(N(BB(N(BBN)N))N)B)B$$
$$H(B(H(HB))B)B(HH)$$

The combinator N may merit further study, for—in addition to offering as much (or more) power as W for constructing fixed point combinators—$BN(BB)$ acts as S does. The significance of this comment rests with the fact that S plays a vital role in the study of combinatory logic as a whole—S and K form a complete axiom system for the logic. Without the Argonne program OTTER and the kernel strategy, the combinator N might have remained undiscovered.

Smullyan then posed corresponding questions about the presence of the strong fixed point property for the fragments with respective bases of combinators B and L and of Q and L, defined by the following equations.

$$Lxy = x(yy)$$
$$Qxyz = y(xz)$$

McCune and I proved that neither fragment could satisfy the strong fixed point property [18]; OTTER again played a key role.

Smullyan also posed questions concerning the existence of *combinations*, an expression involving some given set of combinators that behaves as some other given combinator does. When an appropriate combination exists, he wished also to know the length (in symbol count) of the smallest and how many of such there are. The questions McCune and I answered focus on the following combinators.

(B)	$Bxyz = x(yz)$	(Q)	$Qxyz = y(xz)$
(C)	$Cxyz = xzy$	(Q_1)	$Q_1xyz = x(zy)$
(F)	$Fxyz = zyx$	(T)	$Txy = yx$
(G)	$Gxyzw = xw(yz)$	(V)	$Vxyz = zxy$

From the theorems proved in the study, the following summarizes the important results [39]. The combinators to be used for attempting to find an

appropriate combination are B and T. Two combinations exist that act as Q does, each of length 6, and no shorter exists. Two combinations exist that act as Q_1 does, each of length 6, and no shorter exists. Two combinations exist that act as C does, each of length 8, and no shorter exists. Five combinations exist that act as F does, each of length 8, and no shorter exists. Ten combinations exist that act as V does, each of length 10, and no shorter exists. Five combinations exist that act as G does, each of length 10, and no shorter exists.

Questions of either type—those focusing on fixed point combinators and those focusing on combinations—might prove challenging and useful for evaluating a new reasoning program or a new idea. A full appreciation of the power offered by today's automated reasoning program can be gained by accepting the challenge of attempting to answer the questions just discussed, and to do so unaided by a computer program. What satisfaction you and I might derive—especially me with my experiences at the poker table—from witnessing a person's reaction to accepting the challenge, when said person is one of those who still considers automated reasoning at best to have the significance of a mouse eating a grain of wheat! Since you might be curious, paramodulation was the chosen inference rule for both studies of combinatory logic, and the set of support strategy played a key role. The time required to find fixed point combinators is frequently less than 5 CPU seconds, when the kernel strategy is used. Except for the combinator G, the respective times in CPU seconds for the studies of combinations are 6, 14, 19, 50, and 248. For G, the results were obtained from a Prolog program used in a generate-and-test mode; that Prolog program is *not* a reasoning program. In addition to finding the appropriate combinations for the combinator G, the Prolog program was used to show that no smaller combinations exist. Since attempts with OTTER did not succeed, we have an opportunity for the big stakes players.

And now for the new material gathered during the writing of this article: a possibly new axiom system for sentential calculus, a sharply shorter proof of the completeness of the axiom system proposed by Lukasiewicz (consisting of L1, L2, and L3), and some related items. The following story, showing precisely how the axiom system was discovered, provides powerful evidence of the value of having the program OTTER as a member of a research team. The use of OTTER dramatically reduced the turnaround time and facilitated the immediate verification of a conjecture. The story centers on an episode of temporary miscommunication that was the primary force in the discovery of the axiom set to be introduced. At the same time, the story provides another example of the serendipity of research.

After McCune and I dispatched the 68 theorems suggested by Dana Scott in an email, Scott and I began an email exchange of ideas and results. He played the role of logician, bringing knowledge of various calculi into play; I played the role of computer scientist, supplying knowledge relevant to the varied uses of OTTER. Scott's contributions included suggestions of theorems

for OTTER to try to prove and suggestions of where to find other theorems for OTTER to try to prove. My contributions were to respond to Scott's suggestions by including input files, designed to enable OTTER to perform effectively, and proofs, obtained from OTTER using the various input files. Because the third attempt (at proving the 68 theses) succeeded in proving all 68 and in less than 16 CPU minutes on a SPARCstation, we were off and running.

As indicated earlier, among those 68 theses numbered 4 through 71 (with 1 through 3 being the complete axiom system proposed by Lukasiewicz), the set consisting of theses 18, 35, and 49 can be proved to be another complete axiom system, namely, *FL*. In other words, our success with Scott's original suggestion included a proof of the completeness of the Lukasiewicz system. Therefore, an obvious challenge was to attempt to have OTTER prove the converse, that the system consisting of theses 18, 35, and 49 is complete, where the proof would rest on deducing theses 1, 2, and 3.

It worked, and on the first try; theses 2 and 3 were proved in approximately 2 and 4 CPU seconds, respectively. However, as it turned out, the deduction of the remaining thesis, 1, proved more difficult. In fact, were it not for the ability to run OTTER in the background—permitting me to write this article and simultaneously conduct research—and were it not for successes in the preceding years reached only after substantial CPU time had been used, the attempt to complete the desired proof might have been abandoned. Instead, thesis 1 was proved after 26,600 CPU seconds.

The natural move next was to follow another of Scott's suggestions by studying other known axiom systems, which, as it turned out, led to the earlier-mentioned episode of miscommunication. Buried in the text of one of the papers by Lukasiewicz is the mention of the complete system consisting of theses 19, 37, and 59; no proof of its completeness is given, nor does he give any discussion of the importance of this system. Again, on the first try, OTTER proved it complete in less than 7 CPU seconds, with a 29-step proof in which the known system consisting of theses 18, 35, and 49 is deduced. Flushed with pleasure at this new success, I immediately sent the proof to Scott by email. However, as became clear from Scott's return email, I had miscommunicated—I had left the impression that this system, consisting of theses 19, 37, and 59, was a new axiom system, whose discovery belonged to me and, of course, to OTTER. What to do?—instant notification of my error, which I sent, certainly seemed hardly enough. After all, I had in effect been congratulated—which I obviously did not deserve—and, perhaps equally disturbing, Scott had expressed interest in the axiom system.

What was called for—if it could be found—was a genuinely new axiom system. The preferred way to rectify my communication error was to present—if at all possible—Scott with an axiom system that was genuinely new, *not* already in the literature. Unfortunately, I had no intuitive grasp of the elements of sentential calculus, and no knowledge of how its study had proceeded.

Therefore, on the surface, the idea of making such a search was out of the question—unless a guess and reliance on the ever-present and valued assistant OTTER could save the day.

The morning after the episode of miscommunication, noting that various complete systems existed each consisting of three theses—1 and 2 and 3, 18 and 35 and 49, and 19 and 37 and 59—I read through the 68 originally suggested by Scott. My objective was to replace one of theses 19, 37, and 59, for Scott had liked that system. Influenced by the apparent relation of 37 to 59, the placement and occurrence of their variables and the function n, the thesis that appealed to me above the rest was 60. Therefore, the system to try to prove complete was that consisting of theses 19, 37, and 60. I selected the naive approach of seeking a deduction of thesis 59, since both systems share 19 and 37. Such an approach is naive, for it might actually be easier to deduce another system entirely, say that consisting of theses 18, 35, and 49; I must admit that I did not even consider that possibility.

Again OTTER succeeded on the first try; an 11-step deduction of thesis 59 was obtained in less than 3 CPU seconds. Rather than sending that proof to Scott, I then had OTTER seek a shorter proof, which it found. Here is the 8-step proof that I sent to Scott, found by OTTER in just under 5,000 CPU seconds; a shorter proof exists, which will also be given. As observed earlier, the number in the first field indicates the corresponding position among the retained clauses.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
8 [] P(i(i(i(x,y),z),i(y,z))).
9 [] P(i(i(i(x,y),z),i(n(x),z))).
10 [] P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z))))).
25 [] -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_th_59).
--------------------
34 [hyper,8,1,8] P(i(x,i(y,x))).
40 [hyper,10,1,9] P(i(i(i(x,y),z),i(i(u,z),i(i(x,u),z)))).
59 [hyper,40,1,34] P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))).
63 [hyper,59,1,34] P(i(i(x,i(x,y)),i(z,i(x,y)))).
68 [hyper,63,1,63] P(i(x,i(i(y,i(y,z)),i(y,z)))).
30752 [hyper,68,1,68] P(i(i(x,i(x,y)),i(x,y))).
30758 [hyper,30752,1,34] P(i(x,x)).
30954 [hyper,30758,1,10] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
```

Clause (30954) contradicts clause (25), and the proof is complete.

Scott noted that this new axiom system, in which 59 is replaced by 60, offers less appeal, but the proof of 59 is interesting. He then suggested a way to shorten the proof, which OTTER succeeded in doing. Where the preceding 8-step proof is *organic*, meaning that none of the steps (theses) of the proof contains a subthesis that is true under all assignments of true and false, the following 7-step proof is inorganic.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_th_59).
3 [] P(i(i(i(x,y),z),i(y,z))).
4 [] P(i(i(i(x,y),z),i(n(x),z))).
5 [] P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z))))).
------------------
6 [hyper,3,1,3] P(i(x,i(y,x))).
10 [hyper,5,1,4] P(i(i(i(x,y),z),i(i(u,z),i(i(x,u),z)))).
18 [hyper,10,1,6] P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))).
21 [hyper,18,1,6] P(i(i(x,i(x,y)),i(z,i(x,y)))).
26 [hyper,21,1,6] P(i(x,i(y,y))).
27 [hyper,26,1,26] P(i(x,x)).
33 [hyper,27,1,5] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
```

Clause (33) contradicts clause (2), and the proof is complete.

It appears that the two proofs are the shortest that exist among, respectively, the organic and inorganic. A review of this anecdote suggests correctly how a dialogue between researchers can be sharply enhanced by access to an automated reasoning program of OTTER's type.

The preceding story has an interesting postscript, showing, among other things, that wild guesses coupled with OTTER's power and rapid turnaround can lead to additional advances. Roughly a week after the discovery of the system consisting of theses 19, 37, and 60, Scott suggested yet another possible axiom system. He suggested replacing thesis 60 by the following thesis, to be called 60a.

```
P(i(i(i(i(y,z),i(i(x,y),z)),u),i(i(n(x),z),u))).
```

In approximately 47 CPU seconds, OTTER proved (on the first try) the completeness of Scott's system by deducing thesis 59 with the following 8-step proof.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
8 [] P(i(i(i(x,y),z),i(y,z))).
9 [] P(i(i(i(x,y),z),i(n(x),z))).
10 [] P(i(i(i(i(y,z),i(i(x,y),z)),u),i(i(n(x),z),u))).
25 [] -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_th_59).
--------------------
27 [hyper,8,1,8] P(i(x,i(y,x))).
30 [hyper,9,1,8] P(i(n(i(x,y)),i(y,z))).
35 [hyper,27,1,10] P(i(i(n(x),y),i(z,i(i(u,y),i(i(x,u),y))))).
176 [hyper,35,1,30] P(i(x,i(i(y,i(z,u)),i(i(i(v,z),y),i(z,u))))).
193 [hyper,176,1,176] P(i(i(x,i(y,z)),i(i(i(u,y),x),i(y,z)))).
213 [hyper,193,1,8] P(i(i(i(x,y),i(i(z,y),u)),i(y,u))).
253 [hyper,213,1,10] P(i(x,x)).
273 [hyper,253,1,10] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
```

Clause (273) contradicts clause (25), and the proof is complete.

OTTER also obtained a completeness proof of the Scott system by deducing the three axioms of *FL*; the 33-step proof was obtained in approximately 89 CPU seconds. In approximately 94 CPU seconds, an 11-step deduction of thesis 60 was obtained. Of the completeness proofs focusing on the (independent) Frege system, a 31-step proof was obtained in approximately 2,912 CPU seconds. Hilbert's system was deduced with a 24-step proof in approximately 921 CPU seconds. In approximately 2,677 CPU seconds, the Lukasiewicz axiom system was obtained with a 24-step proof. As is so typical of the experiments presented in this article, all of the completeness proofs for the Scott system were obtained in a single run.

A post postscript: I have just found a 4-step, organic proof of thesis 59 from theses 19, 37, and 60; therefore, earlier, I should have said "it appeared that ..."—this article does indeed have the character of a letter. When I notified Scott of this 4-step proof, his reaction, given by email, was that it might indeed be "a very neat proof that would not be obvious to a human investigator". He explained that it is not particularly easy to do unification in one's head—and is he ever right! (Note 1 is relevant to the preceding.) Here is the proof.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
8 [] P(i(i(i(x,y),z),i(y,z))).
9 [] P(i(i(i(x,y),z),i(n(x),z))).
10 [] P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z))))).
25 [] -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_th_59).
--------------------
33 [hyper,10,1,9] P(i(i(i(x,y),z),i(i(u,z),i(i(x,u),z)))).
44 [hyper,33,1,8] P(i(i(x,i(y,z)),i(i(i(u,y),x),i(y,z)))).
68 [hyper,44,1,8] P(i(i(i(x,y),i(i(z,y),u)),i(y,u))).
99 [hyper,68,1,10] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
```

Clause (99) contradicts clause (25), and the proof is complete.

Of course—in keeping with my personality with which you are so familiar, and because I simply delight in proofs and in successes with OTTER—I would like to include virtually everything learned because of Scott's prompting; I clearly thrive on successful proof finding with OTTER, especially when the impetus is provided by a great scholar. However, since I can hear you giving me the appropriate warning, I shall postpone some items to a later section (Random Notes), omit many, and conclude this section with the promised shorter proof.

The theorem in question asserts that the Lukasiewicz system, consisting of theses 1, 2, and 3, is complete for sentential calculus. Scott informs me that the Lukasiewicz proof, deducing the three axioms of *FL*, is 46 steps in length; using OTTER on the Macintosh, Scott obtained essentially the same proof. The following 30-step proof, deducing the three theses of *FL*, is the shortest

so far obtained; a somewhat different 30-step proof that is not organic has also been found with OTTER.

```
1 [] -P(i(x,y)) | -P(x) | P(y).
4 [] -P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) |
          -P(i(i(n(p),n(q)),i(q,p))) |
          $ANS(step_all_Church_FL_18_35_49).
8 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
9 [] P(i(i(n(x),x),x)).
10 [] P(i(x,i(n(x),y))).
--------------------
34 [hyper,8,1,8] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
35 [hyper,9,1,8] P(i(i(x,y),i(i(n(x),x),y))).
38 [hyper,10,1,8] P(i(i(i(n(x),y),z),i(x,z))).
41 [hyper,34,1,34] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
43 [hyper,34,1,8] P(i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))).
52 [hyper,38,1,34] P(i(i(x,n(y)),i(y,i(x,z)))).
57 [hyper,38,1,9] P(i(x,x)).
59 [hyper,52,1,38] P(i(x,i(y,i(n(x),z)))).
66 [hyper,41,1,35] P(i(i(x,i(n(y),y)),i(i(y,z),i(x,z)))).
67 [hyper,43,1,41] P(i(i(x,i(i(y,z),u)),i(i(y,v),
                       i(x,i(i(v,z),u))))).
103 [hyper,59,1,9] P(i(x,i(n(i(i(n(y),y),y)),z))).
272 [hyper,66,1,103] P(i(i(i(i(n(x),x),x),y),i(z,y))).
303 [hyper,272,1,57] P(i(x,i(i(n(y),y),y))).
310 [hyper,303,1,67] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
317 [hyper,310,1,66] P(i(i(i(i(x,y),y),z),i(i(n(y),x),z))).
335 [hyper,317,1,34] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
346 [hyper,317,1,57] P(i(i(n(x),y),i(i(y,x),x))).
350 [hyper,317,1,38] P(i(i(n(x),n(y)),i(y,x))).
377 [hyper,350,1,38] P(i(x,i(y,x))).
416 [hyper,377,1,8] P(i(i(i(x,y),z),i(y,z))).
508 [hyper,416,1,350] P(i(n(x),i(x,y))).
509 [hyper,416,1,346] P(i(x,i(i(x,y),y))).
593 [hyper,508,1,335] P(i(i(n(x),y),i(n(y),x))).
644 [hyper,593,1,508] P(i(n(i(x,y)),x)).
714 [hyper,509,1,41] P(i(i(x,i(y,z)),i(y,i(x,z)))).
765 [hyper,714,1,8] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
771 [hyper,714,1,346] P(i(i(x,y),i(i(n(y),x),y))).
1755 [hyper,771,1,67] P(i(i(n(x),y),i(i(z,x),i(i(y,z),x)))).
1809 [hyper,1755,1,644] P(i(i(x,i(y,z)),i(i(y,x),i(y,z)))).
1870 [hyper,1809,1,765] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
```

Clauses (377), (1870), and (350), respectively, contradict the literals of clause (4), and the proof is complete.

1.6 Mathematicians and Logicians as Users of a Program: Goal 3

Although a precise estimation of the value of having many mathematicians and logicians as users of automated reasoning programs is essentially impossible, their use would clearly advance the field markedly. A master of some area, group theory for example, who was also quite familiar with the intricacies of the use of OTTER would almost certainly make significant contributions to that area. I must admit that this observation seems difficult to doubt when you review what has occurred in the past decade regarding the answering of open questions from a variety of fields of mathematics and logic. Granted that none of the long-standing deep problems have been solved; however, the investigators could hardly be classed as masters of any of the disciplines from which the questions were taken. I also agree with you that failure would be the likely outcome if an automated reasoning program were asked to prove a randomly selected theorem from a standard textbook. Given these pluses and minuses, the goal of having mathematicians and logicians counted among the users of such programs is indeed an important goal and—how can it be otherwise!—an important component of your and my dream for automated reasoning. Finally, I can report that the current set of those using an automated reasoning program does now include such researchers.

Although I have heard rumors of such uses at the University of Texas and in England—rumors that I have not verified—I prefer to confine my comments to what I know personally. Also, recognizing the lack of a clear distinction between that which is strictly computation and that which contains reasoning steps, I shall, nevertheless, pretend otherwise and cite as my first example John Kalman's research. Kalman is a logician at the University of Auckland in New Zealand. In the late 1970s, he had his own theorem-proving program which he heavily used for his research in equivalential calculus [8]. Prompted by his desire to become familiar with our efforts in the field, Kalman visited Argonne and suggested leaving with us seven theorems to attempt to prove with our program. He did not have to wait too long before seeing results. Indeed, ten minutes after he presented the seven theorems, he had in his hands five proofs obtained with a version of the program that would eventually be called AURA [23]. After tasting this finest of brandies, he was converted, shortly thereafter becoming a constant and active user of our programs—and still is today. So convinced of their value, he has written an extensive workbook for the use of ITP, a book that contains many gems that merit study [9]. He is currently using OTTER to aid his research in group theory, but in the context of divisional calculus.

For some time, copies of OTTER have also been in the hands of Corrado Boehm (combinatory logic) University of Rome, Robert Meyer (relevance logic) Australian National University, R. Goldblatt (mathematics) Wellington University in New Zealand, and Roger Hindley (logic) University College

in Wales. Most recently, we sent a copy of OTTER to Dana Scott (logic) Carnegie Mellon University, which he has successfully used on his Macintosh. Barely a trickle, indeed; but, compared to 1975, a veritable torrent!

On the other hand, recognizing that in some sense we have hardly moved from zero, you might find interest in the following excerpts from two phone conversations I had with Irving Kaplansky. Approximately a decade ago, I called him at the University of Chicago to obtain a possible open question to attack with our program. He supplied such a question, the question mentioned earlier regarding the possible existence of certain finite semigroups; Note 3 might be of interest. During that conversation, when automated theorem proving in general was broached, Kaplansky surprised me with the following comments, which I have just checked in a second phone conversation. "You have a friend in court. Should you continue in your research, the mathematics papers written in 2060 or thereabouts will be at a new level of reading, with much of the current level within reach of a theorem-proving program." In our second conversation, prompted by my wish to clear with him my inclusion of his earlier comments, he and I agreed that progress is indeed painfully slow, as it is with programs to automate translation from one language to another.

The snail travels fast, thinks the defenseless plant with fear; the snail travels slow, observes the cat eager to play. I believe—of course, you know I actually mean that I am certain—the significant point is that, finally, we *are* traveling, making recognizable progress. What will you and I witness before we move elsewhere? I wonder, I await, and, most truthfully—how many know that truth takes on numerous possible values?—I thrive on the expectation.

1.7 New and Old Uses for an Automated Reasoning Program

At this point—taking you up on your comment that you do not mind a little repetition—I shall present a general review of what is possible when using an automated reasoning program, in particular, when using OTTER. At the simplest level, in addition to the familiar use of a program like OTTER for finding proofs, such a program offers unexpected power for proof checking, as the following example shows. If a proof has no steps missing, you can take each of its steps and use weight templates to instruct OTTER to prefer expressions that match the pattern of function symbols in the steps of the proof.

An example that I just ran focuses on the completeness proof for the system consisting of theses 19, 37, and 60 in which an 8-step deduction of thesis 59 is found. The goal is to have OTTER proof check the given proof. Therefore, OTTER was given the instruction to use symbol count to assign priority (by weight) to all clauses other than those that match one of the following eight weight templates.

```
weight_list(pick_and_purge).
weight(P(i(x,i(y,x))), 2).
weight(P(i(i(i(x,y),z),i(i(u,z),i(i(x,u),z)))), 2).
weight(P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))), 2).
weight(P(i(i(x,i(x,y)),i(z,i(x,y)))), 2).
weight(P(i(x,i(i(y,i(y,z)),i(y,z)))), 1).
weight(P(i(i(x,i(x,y)),i(x,y))), 1).
weight(P(i(x,x)), 2).
weight(P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))), 2).
end_of_list.
```

The flags for seeking shorter proofs were turned off. And here comes an illustration of the unexpected power when using OTTER in a proof-checking mode; Note 3 is relevant.

Instead of verifying by "finding" the expected 8-step proof, OTTER found the 7-step deduction of thesis 59, the same 7-step proof given earlier. The explanation rests in part with the fact that weight templates of the kind just given are treated as if all variables are indistinguishable. Since step five, just as step one, of the shorter proof matches in its pattern of the function i the first weight template, it is preferred over clauses that do not match one of the templates. OTTER happened to find the shorter proof first and was prevented from completing the longer proof because the two proofs share the same last step and forward subsumption was in use. Disappointment seems misplaced; after all, the shorter proof is often the more elegant.

On the other hand, were you to insist upon verifying the given 8-step proof, or were you to prefer to avoid proofs that are not organic, OTTER offers most of what is needed. In the first case, if the weight of the fifth and sixth steps is set to 1 rather than to 2, OTTER finds the 8-step proof. If, in addition, back subsumption (which discards any retained clause that is subsumed by a clause retained later) is turned off, and if you instruct OTTER to seek more than one proof, then OTTER finds both proofs.

In the second case, the avoidance of proofs that are less likely to be organic is achieved with demodulation [32]. Specifically, if the following demodulator list is included in the input, where "$T" is interpreted as "true", then all clauses containing as a proper subexpression $i(t,t)$ for some term t are immediately purged.

```
list(demodulators).
(i(i(x,x),y) = discard).
(i(y,i(x,x)) = discard).
(i(discard,x) = discard).
(i(x,discard) = discard).
(n(discard) = discard).
(P(discard) = $T).
end_of_list.
```

An analysis of these demodulators shows that the thesis $i(x, x)$ is not discarded, an important observation, for this thesis often plays a key role in a completeness proof for an axiom system for sentential calculus as well as for many other proofs.

Of course, if you do not have a proof in its entirety to check, but have instead some steps that are conjectured to be useful (as when hints are given to a starred exercise), then the corresponding weight templates can be included. In other words, for those points between the ends of the spectrum focusing, respectively, on proof checking and proof finding, OTTER offers much of what is needed.

Somewhat related is the use of OTTER to monitor progress, its own and that of the researcher; I recommend Note 4. For a sample of this use, you can imagine wishing to prove some specific theorem, but finding it difficult to choose between various possible attacks. With one choice of attack, the conjecture asserts that, if lemmas A, B, and C could be proved, then the theorem would be within reach. With a second choice, the key lemmas appear to be D and E. With a third choice, lemmas F, G, and H are thought to offer what is needed. With OTTER and without much effort, the researcher can pursue simultaneously all three lines of attack. A way to proceed is to represent each of the eight lemmas as a unit clause, negate each, include with each the ANSWER literal with an argument designating which lemma is represented, place the negations in the passive list, and start the run. With Unix, the run can be placed in the background. Then, one simply glances occasionally at the output file to read the occurrences of UNIT CONFLICT.

The ANSWER literal, which is ignored by OTTER when testing for UNIT CONFLICT, is included to facilitate monitoring of the program's progress and of the researcher's chosen attacks. In addition, you can include in the axioms list the disjunction of the appropriate negated lemmas disjoined with an appropriate ANSWER literal and occasionally glance at the output file in search of a desired occurrence of the EMPTY CLAUSE; if the disjunction is placed in the passive list, it will be ignored, thus defeating the purpose of its inclusion. Using a disjunction approach is a means for avoiding the tedious piecing together of the various parts of a proof, and is useful for automatically removing duplicate steps from the various parts; in other words, this ploy returns the union of the various parts.

In contrast to the preceding uses, you can even use OTTER for the mundane task of detecting errors in the input or errors in the axioms, errors such as a simple typo or an incorrect interchange of symbols. The means for such detection is that of reading the retained clauses resulting from using the input that might contain an error—perhaps unsuspected at that. In fact, I recently had such an experience. The run was motivated by yet another study of a possible axiom system. A proof was found more rapidly than I expected. However, rather than satisfaction, you can imagine my reaction when I found that the proof contained the following undesirable clause.

```
P(x).
```

In the study in question, the clause amounts to asserting that everything is true—obvious nonsense! Fortunately, as exhibited in the proofs given earlier, OTTER supplies, for each deduction, the immediate ancestors from which the deduction is obtained. By tracing backward through the ancestry tree, the flaw in the input was easily found.

Even when you are certain that the input is flawless, you might wish to examine OTTER's output to decide whether the chosen approach is satisfactory. In particular, one of OTTER's flags can be used to cause the output file to contain all of the generated clauses, including those that are discarded for some reason. An inspection of the generated clauses can lead to the discovery that much of the CPU time is being consumed on what is clearly fruitless consideration. Especially for the expert in the field from which the problem under study is chosen, such an inspection can prove of immense value; Note 4 is perhaps of interest here. Indeed, it seems to me that, if so many successes can be so readily obtained by someone who is hardly more than conversant with a field, then the possibilities are most promising if the user is instead a master of the discipline under attack.

You can also use OTTER to produce test problems of varying difficulty, problems to be used to evaluate the relative strength of a new program or the potential of a new approach. A graduated set of such problems certainly appears to be one of the useful items to have for research in automated reasoning. Such a set should also prove useful as exercises for a course, a book, or an exam. One means for obtaining the desired set of problems, or at least obtaining some of them, is to have OTTER run well past the CPU time needed to complete some simple task. For example, let the task be to prove theses 39 and 40 from the members of *FL*, 18, 35, and 49. Taken together, theses 39 and 40 can be viewed as asserting, for all x, the equality of $n(n(x))$ and x. OTTER completes the given assignment in less than 4 CPU seconds, after retaining 156 clauses. If you then take advantage of the fact that OTTER's performance is nearly linear with regard to the deduction of conclusions and allow the program to run for a few CPU hours, a proof of the dependence of thesis 21 is also obtained. In approximately 7.4 CPU hours, OTTER produces a 55-step proof of the dependence of thesis 21 on theses 18, 35, and 49.

For those who prefer an approach to automated reasoning based on Prolog technology, this problem appears to offer a small challenge; in particular, a level saturation approach with OTTER obtained an 11-step proof in approximately 9.1 CPU hours. The explanation for finding an 11-step proof rather than the 55-step proof rests with the fact that thesis 21 is dependent on theses 18 and 35 alone; the longer proof relies on the use of thesis 49, thus admitting to the search clauses in which the function n occurs. In fact, 28 of the 55 clauses of the longer proof contain at least one occurrence of n. A bigger challenge for the approach based on Prolog technology focuses on proving thesis 1, the first of the three axioms Lukasiewicz used for sentential calculus, from

theses 18, 35, and 49.

Clearly, a quick review of the various uses covered here—from those directly related to research to those that might well be considered mundane—correctly suggests that I am captivated by experiments with an automated reasoning program. That I find the program OTTER to be a great companion and a valuable assistant is manifestly obvious. Finally, you are definitely warranted in concluding that—rather than the impossibility of an effective automation of logical reasoning, as has too frequently been suggested in the literature—I am building a case for my view that the field has demonstrated in various ways that we have arrived.

1.8 Challenges and Open Questions

My friend and colleague George Robinson asked me on September 15, 1990, whether the field was still as prone to bs as it was when he and I studied automated theorem proving. I told him that all was sharply changed; many researchers now welcomed challenges and the opportunity to attack open questions with an existing program. Therefore, in addition to the challenges presented by the theorems discussed so far in this article, others seem in order, including questions that remain open here in 1990.

Open Question in Robbins Algebra

The following question has been open for decades, receiving attention even from Tarski and his students [6, 27]. Is every Robbins algebra a Boolean algebra? A Robbins algebra is a nonempty set satisfying the following three axioms, expressed in clause notation, in which the function o can be interpreted as plus and the function n as negation.

```
(R1)  EQUAL(o(x,y),o(y,x)).
(R2)  EQUAL(o(o(x,y),z),o(x,o(y,z))).
(R3)  EQUAL(n(o(n(o(x,y)),n(o(x,n(y))))),x).
```

A Boolean algebra is a nonempty set S with two operations, plus and times, and a 0 and a 1. Each operation is commutative, and each distributes over the other. The 1 is a multiplicative identity, and the 0 is an additive identity. In addition, for every x, the negation of x exists with x plus its negation equal to 1 and x times its negation equal to 0. An alternative axiomatization of Boolean algebra consists of (R1), (R2), and Huntington's axiom (H3) [7].

```
(H3)  EQUAL(o(n(o(n(x),y)),n(o(n(x),n(y)))),x).
```

Since it is known that every finite Robbins algebra is Boolean, two choices exist. You can attempt to find an infinite model that satisfies the three axioms for a Robbins algebra but violates a property for a Boolean algebra, or you can attempt to prove that Robbins implies Boolean. If you rely on the

assistance of an automated reasoning program, with the intention of proving that Robbins does *not* imply Boolean, an added challenge is presented by the need to cope with infinite sets. If instead you rely on such a program with the objective of proving that Robbins *does* imply Boolean, then an added challenge focuses on coping with the extremely large number of clauses that will ordinarily be encountered, even if paramodulation is the chosen inference rule.

Open Questions in Combinatory Logic

Combinatory logic presents a number of open questions of which the following offer intrigue. Does the fragment with a basis consisting of the combinators B and S alone satisfy the strong fixed point property? The first two of the following three clauses specify the actions of B and S, respectively; the third clause corresponds to the denial of the strong fixed point property.

```
EQUAL(a(a(a(B,x),y),z),a(x,a(y,z))).
EQUAL(a(a(a(S,x),y),z),a(a(x,z),a(y,z))).
-EQUAL(a(y,f(y)),a(f(y),a(y,f(y)))).
```

If paramodulation is the inference rule of choice, then, of course, you must add a clause for reflexivity of equality, $x = x$. If the inference rule of choice is one of the resolution-based rules, hyperresolution for example, then in addition you must add clauses for symmetry, for transitivity, and for substitutivity in each of the arguments of the function a.

For a similar question, does the fragment with a basis consisting of the combinators B and N_1 satisfy the strong fixed point property? The behavior of the combinator N_1 is given by the following clause.

```
EQUAL(a(a(a(N1,x),y),z),a(a(a(x,y),y),z)).
```

To further encourage research focusing on the automation of model generation, here are two open questions. Does there exist a finite model satisfying the combinators B and L in which the strong fixed point property fails to hold? Does there exist a finite model satisfying the combinators Q and L in which the strong fixed point property fails to hold? That some (possibly infinite) model of the type under discussion in each of the two questions exists follows from results quoted earlier from the research with my colleague McCune [18].

Because of the lack of formal criteria to determine precisely which questions merit the classification of open, I point out that the preceding questions from combinatory logic are my questions. I do not mean to imply that I know the answers, for I do not; nor, as far as I can ascertain, does anyone else.

Challenge Problem from Many-Valued Sentential Calculus

As for a new problem (from among the theorems whose proof is known) to take the place of the problems previously considered tough for an automated reasoning program to solve, the following theorem from many-valued sentential calculus provides an excellent target. The theorem, sometimes known as the fifth conjecture of Lukasiewicz, asserts that, from the first four of the following five clauses, the fifth can be deduced by using condensed detachment.

```
P(i(x,i(y,x))).
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(i(x,y),y),i(i(y,x),x))).
P(i(i(n(x),n(y)),i(y,x))).
P(i(i(i(x,y),i(y,x)),i(y,x))).
```

The problem asks for an approach that enables an automated reasoning program to prove the Lukasiewicz conjecture, using condensed detachment. Two points to keep in mind are (1) a proof of the conjecture has been obtained where the problem was rephrased as one for equality, and (2) McCune and I have a 63-step proof using condensed detachment, a proof obtained by using OTTER in a mode that is in part proof checking and in part proof finding. Our solution, obtained in a style somewhat reminiscent of that discussed earlier in which you have an outline and OTTER fills in the holes, does *not* count, for we relied heavily on the equality-oriented proof that was obtained with OTTER.

Challenge Problems from Sentential Calculus

The last two challenges in this section come from Dana Scott. Both focus on a slightly different notation for sentential calculus, namely, the replacement of $n(x)$ by $i(x, F)$, where F is thought of as meaning false. Scott asks whether, among the following five axioms given in clause form, axioms 1, 2, and 4 are complete for sentential calculus, and whether axioms 1, 4, and 5 are.

```
(axiom 1)  P(i(i(x,y),i(i(y,z),i(x,z)))).
(axiom 2)  P(i(i(i(x,F),x),x)).
(axiom 3)  P(i(x,i(i(x,F),y))).
(axiom 4)  P(i(F,x)).
(axiom 5)  P(i(i(i(x,F),F),x)).
```

Axiom 3 is included because its deduction from either set will establish the corresponding completeness result. As it turns out, the following axiom, 2a, is provable from axioms 1, 2, and 3; it is obviously more general than axiom 2.

```
(axiom 2a)  P(i(i(i(x,y),x),x)).
```

Therefore, the complete axiom system consisting of 1, 2, and 3 is equivalent to that consisting of 1, 2a, and 3.

1.9 Random Notes

For your enjoyment and curiosity, I decided to include the following notes. Some of them express the views we have exchanged; some are simply my unsupported opinions (which you have encouraged me to express). Some of the notes are directly related to the earlier material, some are distantly related, and some are perhaps unrelated.

Note 1. That a number of individuals still maintain the impossibility of the effective automation of logical reasoning attests to resistance—and, perhaps, to fear. No automated reasoning program will ever replace the skilled mathematician or logician. But, in view of the frequency, variety, and ease of success reported here and elsewhere, how can one doubt the value of such a program as an automated reasoning assistant?

As my esteemed colleague Bob Boyer has been heard to say, there is no way to cheat when answering an open question. Since a number of such questions have been answered with the claim that an automated reasoning program played a key role, then the only apparent complaint must rest with the belief that the program did very little. If that is the objection, I recommend that the skeptic review the 4-step proof deducing thesis 59 from theses 19, 37, and 60, coupled with Scott's comment. That proof certainly appears to be an example of a proof that a person might not find.

For a more impressive result, I recommend that the curious examine Steve Winker's original 159-step proof that establishes that the formula XHN provides a complete axiomatization of equivalential calculus [38]. Winker's result refuted a long-standing conjecture, and it was obtained with substantial assistance from an automated reasoning program (AURA) designed by colleagues at Argonne National Laboratory.

For equally impressive results, but ones that may offer even more depth, I recommend that the difficult-to-persuade examine the research (with my colleague McCune) focusing on fixed point problems in combinatory logic [37, 18]. In that context, the kernel strategy, used to find fixed point combinators, is of particular satisfaction; its use provides the most uniform success with randomly selected problems from some area of mathematics or logic.

Note 2. Automated reasoning programs can find proofs that a person, even with training and knowledge, might not find; again, the earlier cited 4-step proof of thesis 59 provides an appropriate example. More obvious is the case in which an open question was answered, where an automated reasoning program played a key role. Such a program makes no tacit assumptions. Such a program nicely complements the expertise that can be brought to a study by the researcher. The objection that one cannot know a priori which inference rule to use seems spurious; in many computing environments, one can simply try a number of runs, simultaneously. When a person, rather than a program, is one's colleague, the preferred attack is often unclear.

On the other hand, the desire for choosing quickly and wisely the best

inference rule and strategy provides an excellent motivation for researchers in our field to propose and test metarules for making such choices. Only, let the testing at least include hard problems and new problems, instead of relying on those that have been repeatedly quoted and that are now easy to solve. Of course, I recommend an occasional attempt to answer an open question; but the tests must focus mainly on known results, for, otherwise, progress cannot be measured. I also give the obvious recommendation that data must be given to support claims, CPU time compared to other programs or approaches, number of conclusions drawn, number retained, and the like.

Note 3. To me, it is clear that the field has matured, at least passing through adolescence. Only a few years ago, what chance would we have had with the 68 theorems suggested by Scott? Before OTTER's arrival, what program could perform almost linearly with respect to drawing conclusions? Who would have guessed that, by 1990, a program would exist that could be effectively used (at one end of the spectrum) for proof checking and (at the other end) for finding a new axiom system?

If the field moves forward just a bit more, such a program might well ease the burden of refereeing for journals of various types—give such a future program the outline supplied in the paper to be refereed, and instruct it to fill in the gaps. For a distantly related example, OTTER found that, among the 68 theses of Lukasiewicz sent by Scott, 12 properly subsumes 11, 18 properly subsumes 17, and 26 properly subsumes 27; a pair of this type might occur in a proof to be refereed. Also, when OTTER was given the obvious eight weight templates and asked recently to proof check the 8-step proof of thesis 59 from theses 19, 37, and 60, the 7-step proof was found first. In other words, OTTER sometimes finds a better proof than the one under study, and sometimes finds a theorem more general than either of two theorems—neither of which subsumes the other—whose negations are included to permit the program to prove both theorems in one run.

Note 4. Conducting research with OTTER is strikingly different from conducting research with a person. With a person, one is ordinarily forced to choose, from among the possible attacks, the one that seems most promising. It is simply too confusing and too arduous to sit back and try to consider two or more paths of inquiry simultaneously. However, with OTTER, you can pursue many attacks in a single run. I certainly took this approach in my studies of the completeness proofs for various axiom systems.

It was simple; you choose the axiom system to be proved complete, place its members in the set of support, and, for each other known complete axiom system, place in the axioms the disjunction of the negations of its members. I think it helps to monitor OTTER's progress: you can place the individual negations—each disjoined with an ANSWER literal designating its role—in the passive list, preventing them from participating except for unit conflict and subsumption. Occasional glances at the output file show how things are progressing. Asking about UNIT CONFLICT shows which members of which

axiom system have been proved, which is where the ANSWER literal comes into play. Asking about EMPTY CLAUSE shows which systems have been deduced or, equivalently, which completeness proofs have been obtained.

The other striking difference focuses on the occurrences of success measured in CPU time or measured in the number of retained clauses. Specifically, proofs often occur in bunches. OTTER often proves some of the theorems under consideration in the first few CPU seconds and with a small number of retained clauses. Then, often, an isolated proof is found after say 20 CPU minutes and the retention of 3,000 clauses. At that point, sometimes, OTTER seems to go off and hide, for glances at the output file in search of new successes from among the targets yield nothing new. However—and I am just beginning to accept this—after a long gap in time and retained clauses, another desired result is often obtained.

For such an example, it suffices to glance at the following results of the successful attempt to prove the dependence of theses 1, 2, 3, 16, 21, 24, 39, and 40 on the set consisting of 18, 35, and 49.

```
UNIT CONFLICT at  2.02 sec  104 [bin,103,13] $ANS(step_th_2).
UNIT CONFLICT at  3.69 sec  144 [bin,143,18] $ANS(step_th_39).
UNIT CONFLICT at  3.76 sec  149 [bin,148,15] $ANS(step_th_16).
UNIT CONFLICT at  3.87 sec  157 [bin,156,19] $ANS(step_th_40).
UNIT CONFLICT at  4.04 sec  162 [bin,161,14] $ANS(step_th_3).
UNIT CONFLICT at 1288.96 sec
                            3211 [bin,3210,16] $ANS(step_th_24).
UNIT CONFLICT at 26600.33 sec
                            8089 [bin,8088,12] $ANS(step_th_1).
UNIT CONFLICT at 26635.09 sec
                            8205 [bin,8204,17] $ANS(step_th_21).
```

Note 5. Using OTTER is, to me, often reminiscent of collaborating with a brilliant colleague whose mental processes are complex and sometimes mysterious. Rather than simple poetry or worse, I am instead referring to the intricacy and interplay of OTTER procedures and options.

For example, let the objective be to find one or more shorter proofs. The obvious approach is to set the flags instructing OTTER to compare derivation lengths and to use back subsumption and thus seek, in a straightforward manner, to measure the lengths of two deductions of the same conclusion, preferring the shorter. The approach is a good one, but far more can be done, and certain hidden complications exist. Perhaps not so obvious, the shorter proof terminating in clause D may be missed because of encountering on the way two deductions of clause C, where the first deduction has shorter length than the second, but where the second deduction contains clauses that can be used repeatedly on the path from C to D. In other words, the shorter subpath on the way to proving a theorem may lead to pursuing a longer path to the finish.

Since the topic of shorter proofs is important—such proofs are often more elegant—I shall simply touch on additional options and other complications that are in part hidden. Rather than instructing OTTER to direct its search based on choosing as the focus of attention the "simplest" clause not yet used, you can instead use level saturation, a breadth-first search. In the context of seeking shorter proofs, the appeal of an approach based on level saturation rests with the correlation between proofs of lower level and shorter proofs. However, you can quickly see that there can exist two proofs with the first having lower level than the second, but with the second being shorter. The level of input clauses is 0, and the level of a deduced clause is one greater than the maximum of the levels of the immediate ancestors used to deduce the clause. A proof can be found at level 4 with a length of 7, in contrast to a second proof of the same result having level 5 and length 5. Therefore, an approach based on level saturation is not guaranteed to find the shortest proof, for, if forward subsumption is used—which is almost always recommended—the second proof can have its last step subsumed, preventing completion. In fact, any form of subsumption, backward or forward, can interfere with finding the shortest proof, for the subsumed clause—although less general than its subsumer—may be the key step in a shorter proof.

Nevertheless, when using the ancestor_subsume flag, to avoid retaining many copies of the same clause, the use of back subsumption is recommended. The clause A ancestor-subsumes the clause B if and only if (1) A properly subsumes B or (2) A and B are identical and the derivation length of A is less than or equal to that of B. Just as there can exist clauses A and B that subsume each other, there can also exist two clauses that ancestor-subsume each other. In either case, if forward subsumption is in use, OTTER retains the earlier copy. Rather than asking OTTER to directly seek shorter proofs by comparing derivation lengths, you might instead prefer to use a level-saturation approach with the ancestor_subsume flag turned off. With that choice, you might well decide to use back subsumption to purge less general clauses in favor of newer and more general clauses. However, you should be warned that the use of back subsumption can, unfortunately, interfere with finding a shorter proof. To see how such interference can occur, you can imagine the deduction of two clauses A and B such that B is deduced after A is deduced and such that B back subsumes A. If in addition the path to A is shorter than the path to B, and if the use of A or B could lead in one step to the completion of the desired proof, then the actions of back subsumption can prevent you from finding the shorter proof.

The use of weighting—which is inescapable with OTTER, for the focus of attention will be based on symbol count or on the input weight templates—can block the discovery of a shorter proof. Indeed, the use of so-called less complex expressions in preference to more complex can force the program to miss a proof that might be shorter though relying on more complex clauses. Perhaps the word "miss" is not the best word in a technical sense; the word

"delay" might be better, delay for an inordinate amount of CPU time, for example, so that the proof is in fact missed.

In a similar way, the inclusion of dependent clauses—although promoting naturalness and perhaps the imitation of the approach a person might take—can also interfere with finding a shorter proof. For example, the successful attempt to prove thesis 21 dependent on theses 18, 35, and 49 produced a 55-step proof in which all three theses were used; in contrast, a second attempt in which thesis 49 was omitted produced an 11-step proof.

Although you might strongly and understandably object, I think we have yet more evidence to support the position that the best approach to the automation of logical reasoning does *not* emphasize the imitation of the way a person works. Naturalness of proof and of approach is fine; but its absence to some degree or to a total degree may be the key to obtaining the desired result. All is okay; better to increase the effectiveness of research by complementing the researcher, without the concern of imitation. I was startled by the obvious appeal—to a surprising number of people—coupled with the misleading nature of a recent talk in which the speaker asserted that the successful dispatching of a challenge problem proposed by Frank Pfennig, though laudable, suffered from the fact that Lukasiewicz (the author of the theorem in question) would never have in effect generated the 6,000,000 conclusions that the program did. I am certain that Lukasiewicz did not and would never take such an approach; but the observation is at best irrelevant, if the object is to evaluate the merit of the attack taken by an automated reasoning program.

Finally, intuition can get in the way of finding the desired shorter proof. For example, intuition might suggest that the thesis $i(x, i(y, x))$ is a key to finding a shorter proof. To impose that intuition on OTTER's search, you can use a weight template to give a low weight (high preference) to the following clause.

```
P(i(x,i(y,x))).
```

OTTER would accept the instruction, emphasizing the role of the preceding clause, which might in turn send the program down a long, long path culminating in a proof. The resulting proof might be far longer than that produced if the clause were simply given a weight based on symbol count or, even further, given a high weight to prevent or delay its use.

Quite different from the direct approach to seeking a shorter proof with OTTER is that in which a number of targets are given simultaneously. For example, if the objective is to prove that some given set of axioms is complete for some calculus, by including in the axioms clauses each of which is the disjunction of the negations of a known set of axioms, OTTER can in effect attempt to use each disjunction as a target. If you also set the proofs limit flag to 0, giving OTTER permission to continue to seek proofs until time or memory is exhausted, then the program may in fact succeed in completing a

number of different and sometimes unrelated proofs. Then, rather than OTTER comparing proof lengths, the user makes the comparison. This approach to seeking a shorter proof can, of course, be combined with the direct approach in which OTTER compares derivation lengths of the same deduction.

Of course, the beauty—some might say difficulty—of using OTTER is captured by the converse; such an emphasis based on the researcher's intuition might be just what is needed to find the shorter proof. Indeed, when Scott suggested that the 8-step proof of thesis 59 from theses 19, 37, and 60 could be shortened and also stated how that could be accomplished, I simply added weight templates to encourage the use of one clause and discourage the use of two others; it did work, as evidenced by the resulting 7-step proof, as he predicted. I view such occurrences as proof that OTTER is an assistant whose value is great—and growing. Conducting research with a person is, in reality, no less complicated.

Note 6. Just as different mathematicians and different logicians have different strengths, so also is it true for automated reasoning programs. No program compares to the Boyer-Moore program for verifying computer code or verifying algorithms. The Kapur-Zhang program appears to be the best for solving problems in which associative/commutative unification can be used. As for the approach based on Prolog technology, Stickel's program sets the standard. However, for my type of experimentation, no program offers nearly as much as OTTER does. Even further, for serious research in some area of mathematics or logic, I strongly conjecture that nothing that currently exists compares to OTTER. The fact that, with respect to drawing conclusions, this program performs almost linearly gives it a distinct edge. Even after 19 CPU hours on a SPARCstation, the rate of deductions had changed only from 550 clauses per CPU second to 460. One of McCune's masterful moves was to adapt the use of discrimination trees [19] to the application of forward subsumption, a procedure that is indispensable for serious research [40].

Sometimes, the objective is to find the "best" proof, which may mean the shortest, the most elegant, the most natural, or simply one that the researcher particularly likes. With the use of weighting and the ancestor flag and demodulation, OTTER often enables one to take a proof and polish it to produce a better proof. A case in point: when Scott suggested that the 8-step proof of thesis 59 from theses 19, 37, and 60 could be shortened to a 7-step proof and showed how, it was trivial to instruct OTTER to follow Scott's suggestions. Later, when the assignment for OTTER was to simply verify that proof to check that no shorter proof existed, the use of level saturation failed in an odd way—a 4-step proof was found instead.

Note 7. Clearly, the use of OTTER as a team member requires reorientation. For one of the more striking examples—when compared to working with another person—no means exists to ask OTTER how it is doing, whether it is close to a proof, or whether it is stuck in a trough. Instead, one notices that the proofs often come in clusters and that the gaps between successes can vary

widely. I have often had the urge to truncate a run because of noting that no new proofs had been found in the preceding 3 CPU hours, only to discover, after resisting the urge, that more proofs would be found. For example, in a run being made as I write this section, in which the object is to find a proof of the completeness of a single axiom (found by Meredith) for sentential calculus, one of the desired results was obtained at the 15 CPU-second mark, and the next result was not obtained until the 5,606 CPU-second mark.

Also, I find it much easier to wander through OTTER's results than through those of a person; the explanation rests in part with no missing steps, precise history of derivation, and the like. Such wanderings have produced unexpected treasure. For example, before I was told that theses 18, 35, and 49 form a complete axiom system for sentential calculus, I was using OTTER to attempt to prove that the system consisting of theses 16, 18, 21, 24, 35, 39, 40, and 49 was complete. The approach focused on deducing theses 1, 2, and 3, the Lukasiewicz axiom system. In that study, I had also instructed OTTER to seek, where possible, shorter proofs. You can imagine the delight and excitement I experienced at discovering, during my examination of OTTER's success with proving theses 1, 2, and 3, that the shorter proofs did not use theses 24, 39, and 40. In other words, I had a proof that theses 24, 39, and 40 were dependent on theses 18, 35, and 49—possibly, although unlikely, a new result. Even though the result was already known, as it turned out, the example clearly illustrates—at least to me—the potential value of OTTER as a research assistant. The fact that theses 16 and 21 are also dependent—although not discovered in my cited experiment—detracts for me in no way. On the contrary, the experiment now gives me yet another way to seek the existence of dependent axioms.

1.10 A Sincere Debate

At this point, before giving a summary and conclusions, an exchange of views is in order. I move that you and I more or less debate. Since you have lost your vote, for you are not present, the motion carries. However, fear not, for I shall also take your part—at least to the extent of guessing what you might say at various points. In fact, I shall give you first say.

"Larry, Larry, you are really confused. There is no way that anyone has come near to fulfilling my dream, not even you, for whom I have the greatest respect. You and your colleagues at Argonne are pretty good at solving some mathematics problems, that's for sure. But so is Macsyma and Wu's algorithm. In all three cases, the problem is that the mainstream of mathematics is just not being addressed. Who cares about little problems like centuries-old geometry theorems, propositional calculus independence results, or tricky little integration problems? Who cares? Not any significant portion of the working mathematicians of this world."

You raise a number of excellent points in your first salvo. I am certain that your sole motive is to get at the truth, so let me address two of those points. You are indeed correct that your grand dream has hardly been touched, but what about those aspects addressed in this article/letter? We can prove theorems suggested by a well-respected researcher, such as Dana Scott, and do so on the first try. A few mathematicians and logicians are now experimenting with an automated reasoning program, OTTER in particular. And—most significant, at least to me, and I hope to others—we have used our programs to answer open questions, and from a variety of fields.

The kernel strategy that McCune and I formulated for answering questions concerning fixed point combinators appears to offer far more power in that area than any person can offer—no disrespect intended to anyone. Who would have thought that by the mid-1980s there would exist an area in which the probability of success was higher than one half when a problem from that area was submitted to an automated reasoning program? And, please note, we usually have the answer in less than 5 CPU seconds. So, we have at least fulfilled one of Ross Overbeek's dreams. Second, as for the mainstream not caring, at least (in alphabetical order) Kalman, Kaplansky, and Scott care some. In fact, Kalman, on a visit to Argonne, told me that the work that Winker and I did added stature to equivalential calculus. I am getting more space than you are.

"Larry, you were a mathematics graduate student at two great schools, the University of Chicago and the University of Illinois. Take down the fifteen or so graduate-level mathematics texts that you studied and mastered. Open those 7,000 pages to any page you want. Is there a program on earth that can check that page, even if it is recast in a formalism of your choice? How many of those pages could be processed? I bet not 10. Quaife's splendid work with OTTER shows that to check regular math (for example, set theory or number theory), you have to put in dozens of lemmas to get through a regular math proof with OTTER, say one lemma per line of proof! That is proof checking, not automated theorem proving. So we are probably less than 10/7000th of the way towards fulfilling my dream. And I will bet you any amount of money that my dream will be fulfilled only when machines do as people do, as I have been saying and trying to do for two decades: use analogy, use a huge database of known mathematics, use examples, use domain-specific heuristics."

You again score points, and in abundance. You see, I focus on what has been accomplished, rather than on what has not. Indeed, an overwhelming majority of what is found in mathematics books is outside our power—now. But, we cannot begin in the middle; mathematics did not begin there. We have come so far from those early days in the 1960s, but the journey has only begun. Only in the past few years have there existed more than a scant number of researchers trying to do mathematics with a reasoning program. Too many people in the field wasted time early—in my judgment—by focusing exclusively on theory rather than experimenting heavily. Granted that much

of Quaife's excellent effort is proof checking; however, a study of it might unearth new ideas to replace much of the checking with proof finding.

I have the greatest respect for your analysis; but, nevertheless, I do not advocate having computers approaching mathematics as people do. Instead, I favor a team consisting of a mathematician and an automated reasoning program—a team that may eventually be far more powerful than either could be! In that regard, although our dreams—yours and mine—diverge sharply, we must each believe that the field clearly merits total devotion to it. Proof: the sum of the years you and I have given to the corresponding research approaches half a century!

"Larry, I hope I have not been too harsh!"

Just harsh enough—not too much, but, more important, not too little. Questioning from those within the field is required—if the field is to advance. However—and I also hope you find my remarks not too harsh—I think that the disappointment and sometimes disapproval regarding the automation of reasoning, especially in the context of mathematics, misses the following fundamental and essential point. In my view, to evaluate the effectiveness of the better automated reasoning programs by comparing them to the best of mathematicians and logicians—for example, those who currently hold the position of full professor and those who in the past have written some of the standard texts—is the wrong comparison to make. If instead the comparison is with people in general, or even with randomly chosen college students, the more effective reasoning programs now available will receive excellent grades, scoring much higher than the vast majority of people. Indeed, given a set of problems from mathematics and logic, the best of such programs—OTTER is my choice—reason far more effectively than almost all people do. For an example, the ability of OTTER to simplify and canonicalize expressions in most cases dwarfs that of any individual I have encountered. OTTER can apply as many as 4,000 demodulators per CPU second; obviously, no person can compete with such performance.

Of course, let us not ignore the long-term goal of using an automated reasoning program that *does* compare favorably with the better mathematicians and logicians. When and if the field has advanced that far, we shall be savoring the finest brandy and perfect raspberries, listening to Mozart and Bix Beiderbecke reborn, and—perhaps more relevant—answering some of the deepest open questions that exist.

Were I not familiar with the twists and turns of the mind—learned mainly at the poker table—I would be sitting at this keyboard experiencing amazement at the lack of use, by mathematicians and logicians, of some automated reasoning program. What fun they are missing by not using this powerful, portable program! How sad, for the use of a program like OTTER by even a handful of mathematicians and logicians would materially increase the rate of advance of automated reasoning and—from what I can tell—also of mathematics and logic. Could it possibly be that they simply do not know what

a valuable assistant for research OTTER is? With OTTER's availability and with access to today's powerful computers, we may have a mystery here, one focusing on how researchers can resist the temptation to add this program to the team. Such users are, in my opinion, one of the key missing pieces to the rapid advance of our field.

The grand objectives are indeed stunning and beautiful—and worth the gamble.

1.11 Summary, Conclusions, and the Future

My congratulations to you, Woody, for winning the 1991 prize for Milestones in Automated Theorem Proving. How poetic it would be if, many years from now, the winner of that prize had found the basis for the corresponding research by reading this article/letter, stimulated by one of the experiments, challenge problems, or open questions presented here! Even further, perhaps this future winner, excited by the demonstration of the marked advance of automated reasoning over the past decade and intrigued by the apparent power of McCune's program OTTER, will have conducted the prizewinning research on a copy of this very program—a copy obtained by anonymous FTP, for example. After all, the challenge problems do provide a wide spectrum of difficulty and are, therefore, useful for testing new programs and evaluating new ideas. Might the impetus come from one of the included problems in which equality plays no role, or might it come from one of the problems in which equality plays the dominant role? I prefer that the impetus be one of the open questions posed here to further stimulate research in automated reasoning and to promote its application to mathematics and logic.

Perhaps, at this very moment, our future prize winner is pursuing one of the three goals on which the organization of this article is based. The pursuit might be of the first goal, which focuses on the ability of an automated reasoning program, when given a theorem to consider, to obtain a proof on the first attempt and, of equal importance, without much guidance from the user. The proviso about the lack of guidance addresses the understandable concern of those who wish or might wish to use such a program, but without the requirement of mastering the various intricacies and subtleties. Perhaps the needed stimulus will be one of the numerous examples given to show that first-attempt success in fact occurs frequently here in 1990.

Instead, the target might be the second goal, which concerns contributions to mathematics and logic resulting from the use of an automated reasoning program. In that case, the impetus for research might be one of the contributions discussed here—a shorter proof than can be found in the literature, an answer to an open question, or the new axiom system. How satisfying that the theorems that are the focus of these contributions and those relevant to the first goal include ones meriting the attention (in alphabetic order) of

Bernays, Church, Frege, Hilbert, Kalman, Kaplansky, Lukasiewicz, Scott, and Smullyan.

Unfortunately, compared to the first two goals, less evidence exists for the realization of the third goal, the use by mathematicians and logicians of an automated reasoning program. However, as such use increases—as it assuredly must—automated reasoning will make impressive advances. I eagerly await the succession of open questions that will be answered with the assistance of some automated reasoning program. Will one of those questions be so significant that answering it will merit the awarding of the $100,000 prize you had established for the "first monumental contribution to mathematics made by an automated reasoning program"?

Has the evidence cited here begun to persuade you that the three goals that are an important part of your grand dream for automated reasoning are in fact finally a reality? The case is further strengthened if you also consider the recent advances in theory, implementation, and application. With regard to theory, the formulation of the kernel strategy [37] for attacking fixed point problems in combinatory logic and the full development of the formal treatment for linked inference [28] in which equality plays no essential role fit nicely into the picture of a field making important advances. As for implementation, the nearly linear performance of OTTER coupled with its power marks a singular development. In the area of application, among the uses for an automated reasoning program discussed in this article are those of systematically seeking shorter proofs, identifying dependent axioms, checking proofs, finding new axiom systems, and the familiar proving theorems. These varied uses make an automated reasoning program—especially the program OTTER—an excellent research assistant. Its power reduces the turnaround time dramatically, producing a qualitative improvement in the study of the chosen area of mathematics or logic. The added power is in part derived from McCune's adaptation of the use of discrimination trees [19] for the application of forward subsumption [21] and demodulation [32].

Of course, as you know so well, I am somewhat puzzled at the lack of use of an automated reasoning program by mathematicians and logicians. Could the explanation be that people simply are not familiar with the recent developments? Is it not known that a program like OTTER can apply as many as 4,000 rewrites rules per CPU second on a SPARCstation to simplify and canonicalize rapidly and accurately? Do few know that, by using weighting [15], the expert's knowledge and intuition can be used to direct such a program in its search for the desired information? More generally, is there little awareness of how well the use of an automated reasoning program complements the way a person works?

In unification alone, this complementary aspect is nicely exemplified. Indeed, as Dana Scott observes, unification is neither the most natural nor the easiest process for a person to apply, which explains in part why the use of a reasoning program can lead to finding unexpected and elegant proofs that

a person might not discover. An excellent example is provided (in Section 5) by the 4-step proof of thesis 59 from theses 19, 37, and 60 (the new axiom system). Dijkstra has commented on the importance of finding such short and elegant proofs [3].

In summary, to me—and I hope to you—the evidence presented in this article strongly suggests that automated reasoning has made significant quantitative and qualitative advances in the decade from 1980 to 1990. Further, this evidence refutes the position that still exists asserting the impossibility of an effective automation of reasoning. Of course, no program will ever replace the mathematician and logician; but programs *do* exist that nicely complement such a researcher. That we have barely begun is clear; that challenges will always exist is preferred; that automated reasoning offers intrigue, excitement, and beauty will continually be true.

Acknowledgments

My gratitude to William McCune for his impressive program OTTER and for being a splendid colleague, and to Dana Scott for his intriguing and excellent suggestions regarding research.

Bibliography

[1] H. P. Barendregt (1981): The Lambda Calculus: Its Syntax and Semantics. Amsterdam: North-Holland.

[2] M. Davis and H. Putnam (1960): A Computing Procedure for Quantification Theory. J. of the ACM 7, 201-215.

[3] E. Dijkstra (1989): Dijkstra's Reply to Comments. In: A Debate on Teaching Computing Science. C. of the ACM 32(12), 1397-1414.

[4] J. Guard, F. Oglesby, J. Bennett, and L. Settle (1969): Semi-automated Mathematics. J. of the ACM 16, 49-62.

[5] P. R. Halmos (1960): Naive Set Theory. Princeton: D. Van Nostrand Co., Inc.

[6] L. Henkin, J. Monk, and A. Tarski (1971): Cylindric Algebras, Part I. Amsterdam: North-Holland.

[7] E. Huntington (1933): New Sets of Independent Postulates for the Algebra of Logic, with Special Reference to Whitehead and Russell's Principia Mathematica. Trans. of the AMS 35, 274-304.

[8] J. Kalman (1978): A Shortest Single Axiom for the Classical Equivalential Calculus. Notre Dame J. Formal Logic 19(1), 141-144.

[9] J. Kalman (1986): An ITP Workbook, Technical Report. ANL-86-56, Argonne National Laboratory, Argonne, Illinois.

[10] J. Lukasiewicz (1963): Elements of Mathematical Logic. Oxford: Pergamon Press.

[11] J. Lukasiewicz (1970): The Equivalential Calculus. In: L. Borkowski, ed.: Jan Lukasiewicz: Selected Works. Amsterdam: North-Holland.

[12] E. Lusk, W. McCune, and R. Overbeek (1982): Logic Machine Architecture: Kernel Functions. In: D. W. Loveland, ed.: Lecture Notes in Computer Science, Vol. 138. New York: Springer-Verlag.

[13] E. Lusk, W. McCune, and R. Overbeek (1982): Logic Machine Architecture: Inference Mechanisms. In: D. W. Loveland, ed.: Lecture Notes in Computer Science, Vol. 138. New York: Springer-Verlag.

[14] E. Lusk and R. McFadden (1987): Using Automated Reasoning Tools: A Study of the Semigroup F2B2. Semigroup Forum 36(1), 75-88.

[15] J. McCharen, R. Overbeek, and L. Wos (1976): Complexity and Related Enhancements for Automated Theorem-Proving Programs. Computers and Mathematics with Applications 2, 1-16.

[16] W. McCune and L. Wos (1987): A Case Study in Automated Theorem Proving: Finding Sages in Combinatory Logic. J. Automated Reasoning 3(1), 91-108.

[17] W. McCune (1990): OTTER 2.0 Users Guide, Technical Report. ANL-90/9, Argonne National Laboratory, Argonne, Illinois.

[18] W. McCune and L. Wos (1989): The Absence and the Presence of Fixed Point Combinators, Mathematics and Computer Science Division Preprint MCS-P87-0689, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois.

[19] W. McCune (1990): Experiments with Discrimination-Tree Indexing and Path Indexing for Term Retrieval. J. Automated Reasoning (to appear).

[20] G. Robinson and L. Wos (1969): Paramodulation and Theorem-Proving in First-Order Theories with Equality. In: B. Meltzer and D. Michie, eds.: Machine Intelligence 4. Edinburgh: Edinburgh University Press.

[21] J. Robinson (1965): A Machine-oriented Logic Based on the Resolution Principle. J. of the ACM, 12, 23-41.

[22] J. Robinson (1988): Automatic Deduction with Hyper-Resolution. International J. Computer Mathematics 1, 227-234.

[23] B. Smith (1988): Reference Manual for the Environmental Theorem Prover, An Incarnation of AURA, Technical Report ANL-88-2, Argonne National Laboratory, Argonne, Illinois.

[24] R. Smullyan (1985): To Mock a Mockingbird. New York: Alfred A. Knopf.

[25] R. Smullyan (1987): Private Correspondence.

[26] R. Statman (1986): Private Correspondence with Raymond Smullyan.

[27] A. Tarski (1980): Private Communication.

[28] R. Veroff and L. Wos (1990): The Linked Inference Principle, I: The Formal Treatment. J. Automated Reasoning (to appear).

[29] S. Winker, L. Wos, and E. Lusk (1981): Semigroups, Antiautomorphisms, and Involutions: A Computer Solution to an Open Problem, I. Mathematics of Computation 37, 533-545.

[30] S. Winker (1982): Generation and Verification of Finite Models and Counterexamples Using an Automated Theorem Prover Answering Two Open Questions. J. of the ACM 29, 273-284.

[31] L. Wos, G. Robinson, and D. Carson (1965): Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. J. of the ACM 12, 536-541.

[32] L. Wos, G. Robinson, D. Carson, and L. Shalla (1967): The Concept of Demodulation in Theorem Proving. J. of the ACM 14, 698-709.

[33] L. Wos and G. Robinson (1973): Maximal Models and Refutation Completeness: Semidecision Procedures in Automatic Theorem Proving. In: W. Boone, F. Cannonito, and R. Lyndon, eds.: Word Problems: Decision Problems and the Burnside Problem in Group Theory. New York: North-Holland.

[34] L. Wos, R. Overbeek, E. Lusk, and J. Boyle (1984): Automated Reasoning: Introduction and Applications. Englewood Cliffs, New Jersey: Prentice-Hall, Englewood Cliffs.

[35] L. Wos, S. Winker, R. Veroff, B. Smith, and L. Henschen (1984): A New Use of an Automated Reasoning Assistant: Open Questions in Equivalential Calculus and the Study of Infinite Domains. Artificial Intelligence 22, 303-356.

[36] L. Wos (1987): Automated Reasoning: 33 Basic Research Problems. Englewood Cliffs, New Jersey: Prentice-Hall,

[37] L. Wos and W. McCune (1988): Searching for Fixed Point Combinators by Using Automated Theorem Proving: A Preliminary Report, Technical Report ANL-88-10, Argonne National Laboratory, Argonne, Illinois.

[38] L. Wos, S. Winker, W. McCune, R. Overbeek, E. Lusk, R. Stevens, and R. Butler (1991): OTTER Experiments Pertinent to CADE-10. Technical Report. ANL-89/36, Argonne National Laboratory, Argonne, Illinois.

[39] L. Wos, S. Winker, W. McCune, R. Overbeek, E. Lusk, R. Stevens, and R. Butler (1990): Automated Reasoning Contributes to Mathematics and Logic. In: Lecture Notes in Artificial Intelligence, Vol. 449. New York: Springer-Verlag.

[40] L. Wos, R. Overbeek, and E. Lusk (1990): Subsumption, A Sometimes Undervalued Procedure. In: J.-L. Lassez, ed.: Festschrift for J. A. Robinson (to appear).