Missing Proofs Found*

Branden Fitelson
University of Wisconsin
Department of Philosophy
Madison, WI 53706
and
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439-4801
email: fitelson@facstaff.wisc.edu

Larry Wos
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439-4801
email: wos@mcs.anl.gov

Abstract

For close to a century, despite the efforts of fine minds that include Hilbert and Ackermann, Lukasiewicz, and Rose and Rosser, various proofs of a number of significant theorems have remained missing—at least not reported in the literature—amply demonstrating the depth of the corresponding problems. The types of such missing proofs are indeed diverse. For one example, a result may be guaranteed provable because of being valid, and yet no proof has been found. For a second example, a theorem may have been proved via metaargument, but the desired axiomatic proof based solely on the use of a given inference rule may have eluded the experts. For a third example, a theorem may have been announced by a master, but no proof was supplied. The finding of missing proofs of the cited types, as well as of other types, is the focus of this article. The means to finding such proofs rests with heavy use of McCune's automated reasoning program OTTER, reliance on a variety of powerful strategies this program offers, and employment of diverse methodologies. Here we present some of our successes and, because it may prove useful for circuit design and program synthesis as well as in the context of mathematics and logic, detail our approach to finding missing proofs. Well-defined and unmet challenges are included.

*This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Keywords: automated reasoning, missing proofs, term-avoidance proofs

1 Challenging Problems, Unaided Researchers, and Automated Reasoning Programs

Known to many, and especially to researchers, is the continual seeking by mathematicians and logicians of sound and rigorous proofs. Although success is frequently the result, sometimes the quarry eludes even great minds such as Hilbert and Ackermann, Tarski and Bernays, Lukasiewicz, and Rose and Rosser. Indeed, rather than the type of axiomatic proof in focus in this article, often such masters have recourse to equality, to substitution, and to other means. In such cases, one cannot say whether the desired axiomatic proof was always in hand. When that occurs, a challenging problem is presented—a desired proof is *missing.*

Less well known, but still vital, is access to sound proofs in the context of verification, both of chips and circuits and of computer programs. Indeed, with the focus on chips free of bugs, such firms as Intel and AMD currently (here in the year 2000) are most concerned with proofs of various theorems. The consideration of models is often found lacking, which explains why such firms are now seriously interested in theorem proving. When an appropriate proof is not in hand, again a desired proof is *missing.*

Proofs come in many forms: constructive, forward, backward, direct, indirect, by contradiction, first-order, higher-order, and, preferred by many among the other types, axiomatic. In addition to their aesthetically pleasing properties, axiomatic proofs offer various advantages, not the least of which is their clarity. We cannot say with certainty what was the primary appeal of such proofs for Hilbert, known to some as Mr. Axiom. However, where automated reasoning is involved, the very nature of axiomatic proofs makes them particularly attractive, especially when one or more designated inference rules are to be used. (In this article, we are not concerned with disproving some conjecture by finding an appropriate model; such a disproof can, of course, be considered a proof, a proof that a result does *not* hold.) Therefore, when the hunt for a missing proof begins in earnest and an automated reasoning program (such as William McCune's OTTER [McCune1989]) is part of the team, the target is indeed an axiomatic proof.

In contrast to the unaided researcher (who has knowledge, experience, and intuition to draw upon), the mere decision to seek an axiomatic proof that is missing is in general of little use to a reasoning program; the problem is too broad. Indeed, in place of what a researcher brings to a problem, the program must be given some guidelines to direct its attack, and the researcher must select the methodology to be employed. (Sufficient for finding various proofs is OTTER's *autonomous mode,* which removes much of the decision making from the researcher; however, here we are concerned with the challenge of finding proofs that

have been missing for decades and with specific guidelines and diverse methodologies that led to diverse successes.)

The choice of the approach aimed at finding a missing proof is often sharply influenced by the type of missing proof. In other words, the problem of seeking a missing proof is replaced by a subproblem, in part delineated by the type of missing proof being sought. In this article we focus on various classes of missing proof (given in Section 2), and we discuss how some of the classes naturally suggest a feasible approach to take.

The most familiar class of missing proofs is encountered when studying an open question offered by some mathematician or logician. Such was the case for more than six decades where the question concerned the possibility that every Robbins algebra is a Boolean algebra. When that question was answered in the affirmative by McCune [McCune1997] using one of his reasoning programs, EQP, it marked a monumental success for automated reasoning. For a second example, we turn to Epstein's recent book [Epstein1994, chapter 2], where he offers two open questions concerning the (in)dependence of two sets of axioms for classical propositional logic. Both of these open questions have been answered using OTTER (unpublished). Indeed, several other open questions in Epstein's book have been partially resolved using OTTER as well. To prove that no dependency exists, one can supply appropriate models. On the other hand, to show that one of the axioms is dependent on the remaining, one can supply a proof that completes with the deduction of the dependent member. OTTER serves well in either capacity. As long as a question is open, a proof is missing. (For additional open questions to consider, see Chapter 11 of [Wos1999].)

In addition to the area focusing on open questions, proofs may be missing for a variety of other reasons. In the worst case, the purported theorem may not, in fact, be a theorem. Of a sharply different nature and a case less highlighted than that for open questions, a proof may exist based on metaargument, but the more preferred axiomatic proof remains missing, one, say, based on the use of the inference rule *condensed detachment*. (The proofs we feature here all rely on that inference rule alone.) Then there is the case in which the proof may be incomplete; indeed, such is often the case with proofs supplied by even well-respected researchers. Or, although various proofs exist, each may be considered unsatisfactory for one or more reasons. For example, one may desire a proof that totally avoids the use of some type of term or that is shorter than all of those offered by the literature.

In this article, we show how missing proofs of diverse types (including those already cited) can be—and indeed have been—found. The key is reliance on McCune's automated reasoning program OTTER. The proofs provided by OTTER have two important properties. First, the proofs are free of error. In mathematics and in logic, ideally, no doubt must exist concerning the accuracy of a proof; indeed, no question must remain concerning the assertion that a given result has in fact been proved. Fortunately, OTTER's proofs are flawless, at least those many we have thoroughly proof checked. Second, the proofs offered by OTTER can be completely verified by using one of its features, set(build_proof_object), followed by

the use of Ivy (written by McCune in the Boyer-Moore logic). In particular, the proofs can be appropriately detailed (including history and term-substitution for variables) by using the option set(build_proof_object).

This article shows that we have had considerable success in using OTTER to find proofs that had either eluded the great minds of some of the masters in logic, or at least had not been reported by them in the literature. In large part, our success rests with the reliance on diverse and powerful strategies this program offers and on the use of various methodologies. Although clearly not an algorithm—almost never for deep problems does an algorithm exist—the methodologies we offer can easily be applied by other researchers.

Clearly, the finding of missing proofs is of substantial interest to mathematicians and logicians. Less obvious is the fact that proofs are of substantial interest to other disciplines, such as circuit design and computer programming. Indeed, an unexpectedly elegant constructive proof can point the way to the design of a far more efficient circuit or present computer code far more effective than was already in hand. For example, a proof that avoids the use of some type of term may provide the key to avoiding the use of an expensive component or the key to sharply reducing the CPU time for a subroutine.

To set the stage, we first briefly discuss some of the types of missing proof OTTER finds (in Section 2.1–2.12). We then (in Sections 3 and 4) provide much detail concerning strategy and methodology, give actual successes, and focus on (what we believe are) startling results. We also include well-defined challenges for researchers to address.

As this article proceeds, the sharp difference between researcher and reasoning program will become evident, as will the power offered when researcher and program perform as a team. The unaided researcher attempting to find a missing proof in general has the likelihood of success increased if few, if any, constraints are imposed. In sharp contrast, when an automated reasoning program is part of the research team, the imposition of constraints often adds to the likelihood of finding a missing proof. Indeed, although perhaps counterintuitive because of (apparently) replacing one goal by a goal that is harder to reach, an added requirement can be the key to success, a requirement such as, say, that the sought-after proof be free of terms of the form $n(n(t))$ for any term t, where the function n denotes negation. In other words, more freedom is good for the unaided researcher, and less freedom is good for an automated reasoning program. Here we show how this aphorism is applied in the context of classifying types of missing proof and then finding such proofs.

2 Missing Proofs of Many Types

An effective approach to attacking a challenging problem presenting wide scope is to replace the general problem by a set of subproblems. By doing so, often a much higher probability exists for increasing understanding, adding to insight, and finding a means to solve the

basic problem. In this section, we adopt such an approach. Indeed, rather than focusing on specific properties and studying the general problem of finding proofs missing from the literature, we instead study a set of subproblems, where each subproblem is concerned with a type of missing proof.

2.1 Term-Avoidance Proofs

A glance at the literature of formal logic shows that, when negation (denoted by n) is part of the theory, terms of the form $n(n(t))$ for some term t are omnipresent in the various proofs that are offered. Is it correct to assume that such double-negation terms are in general indispensable? Put a bit more formally, if T of the form P implies Q is a theorem such that all known proofs of T rely on the use of double-negation, and if both P and Q are free of double negation terms, does there exist a proof of T none of whose deduced steps contains a double-negation term? If so, a type of missing proof can be sought, a proof in the class that can be termed *term avoidance.* The preceding suggests a deep question in logic that merits consideration. Under what conditions, satisfied by P, Q, and the logical system L (that is relevant to the theorem under study) is there guaranteed to exist a double-negation-free proof of Q from P in L, where some of the members of P focus on negation and where all of the members of P participate in the deduction of Q?

A second example of a term-avoidance proof is that in which no deduced step contains as a subterm a term of the form $i(t,t)$ for any term t, where i denotes implication.

OTTER offers via the procedure demodulation and the subtautology strategy just what is needed to seek, respectively, either example of a term-avoidance proof; see Section 3.1 for further discussion and for a set of the appropriate demodulators. Where the parallel for circuit design is the seeking of a circuit in which no logical **or** gates are used, OTTER would thus prove useful. A similar observation holds for program synthesis.

If one takes a much harder look at the literature, one can get the impression that term avoidance in the context of double negation is out of reach, especially for deep theorems. In this article, we focus on proofs that are free of double negation, and we conjecture that our successes (in that regard) may refute an implicit view shared by various logicians concerning how proofs must proceed. In other words, we refute the implicit conjecture (if such is the case) that the reason double-negation-free proofs are frequently *missing* is that they cannot be completed, that they do not exist.

For the key example that refutes the cited implicit conjecture, we focus heavily in this article on a theorem from infinite-valued (or many-valued) sentential calculus. (Many other examples, as well as numerous proofs that are new, will be offered in the planned book *Automated Reasoning and Finding Missing and Elegant Proofs in Formal Logic.*) The specific theorem of interest focuses on the Lukasiewicz five-axiom system for this area of

logic [Lukasiewicz1970]. Intriguing and perhaps most unexpected, not all five axioms are required; indeed, Meredith proves [Meredith1958] that the axiom we call $MV5$ is dependent on the other four.

The literature suggests that, to prove the cited dependence, one must rely on double negation. For example, Meredith's (in effect) 37-step proof (concerning axiom dependence) contains two steps with a term of the form $n(n(t))$ for some term t. (We may have been generous in citing the Meredith proof to be of length 37; indeed, he relies on various lemmas, proved by Rose and Rosser [Rose1958], and we have deliberately assumed that elegant proofs of such lemmas were in hand, which may not have been the case.) Later in this article, we offer a proof that is not only shorter than Meredith's, but that is also free of double negation and that depends on only one of the three key lemmas his proof relies upon. Whether that lemma, denoted by (2.22), is indispensable for a proof of $MV5$ from $MV1$ through $MV4$ is currently unknown to us; it may indeed be an open question.

Incidentally, the fact that just two of Meredith's proof steps involve double negation does not imply that their removal is trivial. For a numerically more impressive result, the original (in effect) 41-step proof showing that Meredith's single axiom axiomatizes all of two-valued sentential calculus contains 17 steps relying on double negation. In contrast, the use of OTTER, its strategies, and various methodologies yielded a proof totally free of double negation.

The proof OTTER found—with the term-avoidance property—establishing $MV5$ to be dependent is of impressively small length and formula complexity. Its length is 32 (applications of condensed detachment), and its formula complexity is 17, meaning that the longest formula in the proof consists of 17 symbols (not counting predicate symbol, parentheses, and commas). By way of comparison, the Meredith proof has length 37 and formula complexity 23. The levels of the 32-step and 37-step proofs are, respectively, 20 and 19, where the level measures the depth of the proof tree. Later in this article, we provide some clues about how the 32-step proof was obtained; far more detail is provided in the already-cited planned book.

2.2 Lemma-Avoidance Proofs

In contrast to the avoidance of some type of term (such as double negation), one might wish to avoid the use of entire formulas or equations. The analogue for programming might be that of avoiding the use of an entire subroutine. If one examines the appropriate literature, for example, that of Rose and Rosser and that of Meredith, one gains the impression that certain lemmas play a key role, that such lemmas may in fact be indispensable. If one could show that such lemmas are in fact not needed, then one would have an illustration of what might be termed a lemma-avoidance proof. Just as missing term-avoidance proofs can be sought, so also can one seek missing proofs of the lemma-avoidance type. Indeed, for a fine

example of such lemma-avoidance proofs, one can seek a circle of pure proofs, say, in the context of a set of equivalent properties. By way of illustration, there exist four defining equations for Moufang loops, each one of which is sufficient, and all four are equivalent. A circle of pure proofs in this context [Wos1996] consists of four proofs for some ordering of the four equations (1 through 4) such that the first proof shows that 1 implies 2 without reliance on 3 or 4, the second shows that 2 implies 3 without reliance on 1 or 4, and similarly for the remaining two proofs.

Such proofs, in the case we study, were missing implicitly, so it appears. Indeed, the literature that we searched repeatedly relies on certain lemmas (such as (2.22), (3.5), and (3.51) [Rose1958].) Lemma-avoidance proofs are also in focus in this article, and we show how (in some interesting cases) an automated reasoning program was used to complete such proofs. Again, one of the keys is demodulation, but the weighting strategy can also be used. Lemma-avoidance proofs may serve well the circuit designer or the computer programmer.

2.3 Metaargument Proofs

Hilbert strongly advocated axiomatic proofs. In his book with Ackermann [Hilbert1950], one finds various proofs in which condensed detachment plays the role of inference rule. However, in that book, two associative laws for the (defined) conjunction and disjunction connectives of two-valued sentential logic are proved, neither via a purely axiomatic approach. Instead, metaarguments are used. In particular, metatheorems concerning the substitutivity of various classes of formulas were proved, and then equality reasoning was used (in addition to the use of detachment). Therefore, we chose to have OTTER assist us in finding the desired axiomatic proofs.

Browsing in the literature sometimes leads to the discovery that many axiomatic proofs are explicitly missing. For example, in [Rose1958], Rose and Rosser (as is true of Hilbert and Ackermann) occasionally resort to metaarguments rather than the explicit use of the axioms. More generally, a paper (such as that cited) can contain proofs that depend solely on condensed detachment, but also contain proofs that depend on that inference rule coupled with metaarguments or depend solely on metaarguments. Far more satisfying, depending on the area of study, are axiomatic proofs that rely on one or more specific inference rules. For example, proofs that rely on condensed detachment alone are usually easier to follow and often far more informative (although, more than occasionally, more difficult to complete).

In this article, we focus on certain laws that play a crucial role in various areas of logic, lemmas whose proof is via metaargument, implying that a condensed detachment proof was possibly unknown and perhaps absent from the literature. We show how OTTER was used to produce the preferred proof. Among the more satisfying, we give such proofs for two cited associative laws of Hilbert and Ackermann. Access to such condensed-detachment proofs may be of especial interest to researchers in logic who are familiar with the fact that,

until now, such proofs have been missing.

An excellent example of what we have in mind is Wajsberg's proof by inductive metaargument [Wajsberg1977] of a generalization of the Tarski-Bernays axioms for the implicational fragment for two-valued sentential calculus. In contrast, we have, because of OTTER, an axiomatic proof relying solely on condensed detachment. One of its charming properties is its length, merely twelve applications of condensed detachment.

2.4 More General Proofs

One of the powerful features of the typical automated reasoning program is its emphasis on generality. The procedure *unification* (which underlies its reasoning and other procedures) seeks the most general substitution for variables to make two expressions share a common domain. Also promoting generality is the use of *subsumption,* a procedure that discards instances of retained information. Because of this generality, the proofs it yields often contain steps that are more general, and often more powerful, than found in the corresponding proof in the literature. Indeed, depending on the significance of the step, a more general step may correspond to a more general lemma that might have been unknown; see the next subsection. By emphasizing generality of deductions, an automated reasoning program greatly enhances its performance.

When all of the proofs of some specific lemma or theorem that are known fail to contain such more-general steps, but such a proof is completable, then, implicitly, a proof is missing. We give examples of this occurrence and note its relevance to the next subsection.

2.5 More General Lemmas, Laws, and Theorems

Because of the generality of the basic mechanism (unification) relied upon by an automated reasoning program (fully detailed in the book [Wos1999]), OTTER occasionally proves more than is expected. In particular, sometimes when the assignment is to prove a specific lemma, OTTER proves a more powerful result. If that result cannot be found in the literature, another type of missing proof is encountered.

In that regard, we present our attack on an associative law, proved by Hilbert and Ackermann, and show how our program proved this law. However, rather than simply finding the desired proof, OTTER proved a more general result. We cannot say at this time (here in the year 2000) whether this more general result is significant in the context of logic.

2.6 Proofs of Unsatisfying Formula Complexity

When a paper contains extremely complex formulas or equations, warranted is the conclusion that a type of proof may be missing. That type is a proof with (perhaps sharply) reduced formula complexity, one that relies on steps that are less complex than offered by the proof in hand.

For our example, we turn to Meredith [Meredith1953]. Meredith presents a short single axiom for the (C,O) representation of two-valued sentential calculus, and he reports a completeness proof. We present (at the close of Section 4) a proof that is considerably less complex—in many quantifiable respects—than the proof he reports. In particular, our OTTER proof illustrates how a reasoning program can be used to find proofs of lesser formula complexity.

2.7 Proofs of Unsatisfying Length

When one encounters a proof of moderate to great length, the natural suspicion is that a far shorter proof may exist, that (perhaps) a proof is missing. The question to be answered concerns how to find shorter proofs. In this article, we give various methods for reaching that objective. We also give some (to us) startling examples of far shorter proofs than found in the literature. (The finding of shorter proofs sometimes directly correlates to the finding of a simpler circuit or more effective bit of computer code, which suggests that the corresponding methodology may be of interest in either context.)

2.8 Omitted Proofs

Authors often omit proofs, for a variety of reasons. Sometimes, the proof is considered obvious; sometimes, the reason for the omission is obscure to the reader. Of course the proof is missing explicitly, missing because the author preferred not to include it.

In any case, one should not conclude (for such missing proofs) that the theorem in question is of little interest or lacks depth. Quite the contrary, as one finds by browsing in the literature of logic [Lukasiewic1970,Meredith1963,Prior1960,Wajsberg1977] and in the literature of mathematics, sometimes with a statement to the effect that various earlier lemmas offer most of what is needed.

In this article, we focus on this class of missing proofs, give examples of proofs we have found (where the proof was omitted), and supply the methodology that was used to fill the void. In that regard, a theorem focusing on a single axiom (of Lukasiewicz) for all of two-valued sentential calculus provides a fine example. As part of this specific topic, we also discuss completing proofs that are far from complete, that contain gaps

in their presentation. A theorem of Lukasiewicz (in which he derives the Tarski-Bernays system from the shortest possible single axiom for the implicational fragment of two-valued sentential calculus) provides an excellent example, one in which steps relying on condensed detachment are missing [Lukasiewicz1970].

2.9 Valid but Unproved Results

In various areas of logic, the establishment of validity (truth in all models) for some result implies that the result is provable. However, without an explicit proof, one is often left with disappointment and, sometimes, curiosity about the missing proof.

Such is the case for two distributive laws of infinite-valued logic that we study. Each has been shown to be valid, but for neither did a proof exist—or so it was the case until we attacked the two theorems with OTTER.

2.10 Distinct-Variable Complexity

Another measure of the complexity of a proof is that focusing on the number of distinct variables required to complete the proof. If a proof is supplied by a master in the field, one might naturally conjecture that its distinct-variable complexity is the best that can be found. If the conjecture has not been proved, then perhaps a type of missing proof has been encountered.

In this article, we give examples of such missing proofs, proofs that, in the context of distinct variables, are better than found in the literature.

2.11 Level Complexity

A still different measure of complexity is that concerned with the level of a proof. The level of a proof step is 0 if the step is taken from the input (from the hypotheses) and is n greater than 0 when one of its parents has level $n - 1$. The level of a proof is the maximum of the levels of its proof steps. The unaided researcher naturally and understandably gravitates toward consideration of lower levels in preference to higher. However, occasionally, the proofs of a given theorem offered by the literature all have a level higher than need be. When such is the case, or suspected to be the case, another type of missing proof has been identified.

In this article, we discuss means that can be used to (sometimes) seek proofs of minimal level.

2.12 Proofs Not Fully Automated

The type of missing proof in focus in this section is missing because of considerations germane only to automated reasoning. Nevertheless, such missing proofs merit study. In particular, when the only proofs of a given theorem that have been completed by a reasoning program required guidance from the researcher, then a fully automated proof is missing.

The pursuit of such a missing proof is merited because, occasionally, the result is the formulation of a new and useful methodology for finding missing proofs of other types. Sometimes, directly or indirectly, the hunt yields a new proof of the theorem in focus. We have indeed experienced this phenomenon. For example, Deepak Kapur used the Meredith proof of his single axiom (for two-valued sentential calculus) to find a six-variable proof, and we in turn used his proof to find an even shorter six-variable proof.

3 Strategy and Methodology for Finding Missing Proofs

Rather than a narrative even vaguely resembling a historical account, here we present approaches that offer a high probability of reaching the objective. We in fact apply these approaches in our current research, relying on OTTER's arsenal of weapons. We shall, in this section, discuss the classes of missing proof in the order found in the preceding section, beginning with term-avoidance proofs that were missing and found with the use of OTTER. Note that common to our attack on various classes of missing proof is an emphasis on an instance of term avoidance, namely, the avoidance of double-negation terms, those of the form $n(n(t))$ for some term t, where the function n denotes logical negation. This move, though counterintuitive, is just the type of action that benefits a reasoning program but that (most likely) hinders an unaided researcher. Briefly (for the curious), we conjecture that the benefit derives from the side effect of increasing the density of useful conclusions among the total that are retained.

For us, the seeking of a missing proof differs little from an attempt to answer an open question. For example, if we find a proof shorter than any that had been known, we treat the result in the spirit that we do when answering an open question that was so classified in the literature.. When we are unable to show that no shorter proof exists than the new proof we have found, (in effect) a new open question has been posed. Of course, some open questions are far more significant than others and are far more difficult to answer. Nevertheless, in general, open questions merit study both for the possibility of finding an answer and for the possibility that the effort may yield an advance for the automation of logical reasoning. The accrual of such advances has obvious practical consequences, not the least of which is an increase in the arsenal of weapons for attacking problems from mathematics, logic, circuit design, program verification, and other disciplines that rely on flawless reasoning.

Regarding the various strategies and methodologies pertinent to our successes, the following reminders, comments, and examples nicely set the stage and may provide some useful hints for other researchers. First, for the set of support strategy, we typically place the axiom or axioms of the logic under study in the corresponding list and place the clause for condensed detachment in list(usable). After all, typically the theorem under attack in the studies reported here offers no special hypothesis, and its denial is best used only to detect proof completion.

Second, regarding the hot list strategy (whose use enables the program to visit and revisit certain chosen items repeatedly before resuming the main attack), we typically place in the corresponding list the axiom or axioms and the clause for condensed detachment. When the research concerns a single axiom, we frequently assign the heat parameter a value of 6 or greater, choosing to do so to cause the program to heavily emphasize and recursively (to the level of 6) visit the single axiom. The use of the hot list strategy can, unfortunately, hinder as well as aid the discovery of a sought-after proof. We currently can offer little evidence as to which will occur.

Third, in the context of the resonance strategy (whose use provides an opportunity for the researcher to suggest formulas or equations that, by their respective patterns, direct the program's attack), we give two illustrations taken from our studies. In the first, where no proof was available (as was the case with the Lukasiewicz 23-letter single axiom), we found profitable for the choice of resonators (patterns to direct the search) formulas corresponding to lemmas (theses) that Lukasiewicz proved in his various publications. If such are not in hand, we recommend for resonators the correspondents of lemmas from a study somewhat related to that from which the theorem under consideration is taken. For our attempt to find the missing proof for the 23-letter single axiom, we included 68 resonators, those corresponding to theses 4 through 71 (cited by Lukasiewicz). In contrast and as a second illustration, we focus on our study of associativity in infinite-valued sentential calculus. Two formulas were to be deduced, whose respective negations are given with the following two clauses.

```
-P(i(i(i(a,i(i(b,c),c)),i(i(b,c),c)),i(i(i(i(a,b),b),c),c))).
-P(i(i(i(i(i(a,b),b),c),c),i(i(a,i(i(b,c),c)),i(i(b,c),c)))).
```

The second target was reached in less than 6 CPU-hours of OTTER's time, on a Solaris workstation (Ultra-2 Enterprise Server). The first target, however, was not reached in more than 10 CPU-hours, with the exhaustion of 240 megabytes of memory. No resonators were used in either attempt. In a second try at reaching the first target, resonators were adjoined, those corresponding to the proof steps that OTTER supplied in reaching the second target, and success was ours (in less than 3 CPU-minutes).

3.1 Seeking Term-Avoidance Proofs

In addition to reliance on the set of support strategy, the hot list strategy, the resonance strategy, and other strategies, the seeking of term-avoidance proofs emphasizes (in our attack) the use of demodulation and, rarely, that of weighting. That emphasis is explained by the fact that, although we know of no way to avoid the deduction of conclusions with some type of unwanted term, OTTER does offer the means for avoiding their retention. Indeed, the approach we take for finding term-avoidance proofs is simply that of discarding conclusions that are deemed undesirable. Depending on the type of term to be avoided, we choose a subset of the following demodulators, with possible modifications based on the nature of the problem to be solved.

```
%  (n(n(n(x))) = junk).
(n(n(x)) = junk).
%  (i(i(x,x),y) = junk).
%  (i(y,i(x,x)) = junk).
%  (i(n(i(x,x)),y) = junk).
%  (i(y,n(i(x,x))) = junk).
(i(x,junk) = junk).
(i(junk,x) = junk).
(n(junk) = junk).
(P(junk) = $T).
```

The use of the appropriate demodulators immediately enables the program to discard conclusions that contain an unwanted term. As an important side effect, (we are fairly certain that) the density of significant conclusions within the total set that are retained is sharply increased.

3.2 Seeking Lemma-Avoidance Proofs

When the goal is to avoid the use of some given lemma to show that a missing proof can in fact be found, a proof in which the lemma is not needed, demodulation again is our choice. For example, if we wish to avoid a lemma, such as that called (3.51) by Rose and Rosser, we include the following demodulator, as well as the appropriate members of the set given in the preceding section.

```
(P(i(i(i(x,y),i(x,z)),i(i(y,x),i(y,z)))) = junk).
```

If OTTER deduces the unwanted lemma, the corresponding demodulator immediately rewrites it to "junk", and almost immediately (with the aid of other demodulators), it is

purged. Because of the nature of demodulation, not only is the unwanted lemma avoided, but also avoided are instances of it (obtainable by instantiation).

3.3 Replacing Metaargument Proofs

Proofs based on metaargument or metatheory are for us and various others before us (such as Hilbert) not the most satisfying. Rather, axiomatic proofs are the choice. The very nature of automated reasoning makes its use a fine weapon for seeking such proofs; indeed, proofs based on metaargument are often out of reach of a reasoning program of the type in focus.

Therefore, when the proof that is missing is of the axiomatic type, our approach is (at the simplest) to submit the corresponding question to OTTER. We in general do not consult the proof based on metaargument, if such is available, relying instead on the fact that a reasoning program of the type under discussion necessarily seeks axiomatic proofs. More precisely, we heavily rely on various strategies offered by this program, prominent among which are the set of support strategy, the resonance strategy, and the hot list strategy.

3.4 Seeking More General Proofs

Mathematics and logic often, but not always, prefer proofs that rely on most general steps, some that are classed as lemmas. Although hardly obvious, the reason that generality is not always preferred rests with the ease of understanding and the (in some cases) unnaturalness of the more general step. If one browses in the literature, one can learn that the approach taken by an unaided researcher often differs sharply from that taken by an automated reasoning program—the latter does not rely on, for example, instantiation.

Further, the nature of unification (which is at the heart of so much of automated reasoning), as well as the nature of subsumption, emphasizes generality. In fact, perhaps amusing and sometimes startling, an attempt at what might be thought of as a straightforward proofchecking assignment often fails because one or more of the steps of the proof under scrutiny is too specific. In place of the too-specific step, the program deduces a more general conclusion due to the nature of unification. When such happens (one or more times), rather than a duplication of the proof in hand, a missing proof can be found, one that relies on more general conclusions.

3.5 Seeking More General Results

The finding of a more general result—law, lemma, or theorem—than that offered by the literature is often most rewarding. Such an occurrence can result from accident or from

a deliberate attempt to do so. The use of an automated reasoning program as a member of the research team increases the likelihood of such a discovery, in part because of the program's emphasis (through reliance on unification) on generality.

We in fact did find a generalization of one half of the associativity law in infinite-valued sentential calculus by accident. The discovery was made indirectly because of our seeking of a shorter proof than we had in hand. Our fortune naturally led us to the deliberate search for the corresponding generalization of the other half of associativity, which proved far more difficult. When we succeeded, we had a pleasing examples of both occurrences of the use of a reasoning program to find a more general result, one by accident, one by intention.

We found the more general form of the other half of associativity (which we then proved does hold) simply by inspection. In particular, we compared the more general form of the second half with the original form and posited its correspondent for the first half.

3.6 Seeking Proofs of Lesser Formula-Length Complexity

The measure of complexity in this section focuses on the number of symbols (excluding parentheses and commas) that occur in the deduced formulas (or equations) of the proof under discussion. If the goal is that of finding a missing proof whose complexity is less than that of the proofs in hand, where the concern is the longest deduced step, OTTER offers precisely what is needed. Specifically, one can instruct the program to "weigh" each deduced conclusion and discard those that exceed a user-assigned value. The weighing can be solely in terms of symbol count, or it can be in terms of weight templates supplied by the researcher. The researcher can assign a value to max_weight to prevent the retention of any new conclusions whose weight exceeds the max_weight, and the max_weight can be chosen to be strictly less than that of any proof known to the researcher.

Failure to find such a missing proof does not preclude its existence. Various reasons account for this misfortune, including the fact that a reduction in formula complexity may force the newer proof to be much longer, which may result in the needed search to be impractical with respect to CPU time.

3.7 Seeking Shorter Proofs

The path to finding shorter proofs than those available is far more complex than it might appear. Indeed, although OTTER offers a mechanism (*ancestor subsumption*) that compares derivation paths that end with the same conclusion and prefers the copy of the conclusion that is reached by the strictly shorter path, its use (not obviously) does not guarantee that shorter proofs will be found. The pitfall and subtlety rest with the simple fact that the shorter proof of an intermediate step may in fact, when relied upon, lead to a longer proof

of the desired final result. Briefly, the explanation is that often the steps of the shorter subproof are of little or no use later in the total proof. In other words, shorter subproofs do not necessarily a shorter total proof make. Nevertheless, the use of ancestor subsumption is one of the arsenal of weapons we typically use when seeking shorter proofs.

Those who hazard that a breadth-first search will solve the problem should be warned that such is not practical. If such a search is made without the use of subsumption to discard types of redundant information, the program will ordinarily drown in new conclusions. If subsumption is employed (as we always do, at least in regard to forward subsumption), then a shorter proof can be missed with a breadth-first search.

We have developed a rather complicated methodology (technically, a set of methodologies) for finding shorter proofs, one in which demodulation, the resonance strategy, and the hot list strategy, among other mechanisms, are relied upon.

By way of the smallest taste of one of the methodologies, note that we take the steps of a proof in hand and use their correspondents as resonators. With demodulation, we then block the use of each, one step at a time. This action forces the program to seek a proof different from that which is known. When we reach a cul de sac, where no shorter proof is yielded, we often relax the constraints, relying on a more generous value for max_weight and for max_distinct_vars and even avoid the use of those demodulators that led to our current best proof.

3.8 Seeking Omitted Proofs

When (in a paper or book) a result of particular significance is announced without proof, the problem of finding the missing proof is especially appealing. The appeal rests in part with the contrast with other types of missing proof, for example, with the class concerned with finding a shorter proof. Indeed, when asked to find, say, a shorter proof, one can use the steps of a known proof as resonators, as patterns to guide the program's search. This aspect applies to many other classes of missing proof. However, when the proof to be found is such that no clue exists, for example, regarding the target, which axioms are to be used, and which lemmas merit attention, the resulting total darkness sharply increases the difficulty but also the intrigue.

In Section 4 of an earlier paper in this special issue [Wos2000], a methodology was presented for seeking hard-to-find proofs. Although the cited methodology was developed to address the type of missing proof discussed in Section 2.12 of this article, it has proved useful in the context in focus in this section. Among its key aspects are the use of the resonance strategy—even though no steps are available to be borrowed from an existing proof—use of lemma adjunction, the set of support strategy, and the hot list strategy. For resonators, we sometimes simply use the correspondents of the proof of some distantly

related result or the correspondents of lemmas proved by the author in question.

3.9 Replacing Proofs Based on a Validity Argument

The knowledge that a result holds because of establishing it to be valid (to be true in all models) in general offers no insight concerning a possible axiomatic proof and not much satisfaction. If a validity argument is all there is, then again (as in the preceding section) no clue exists concerning how to proceed to find the missing proof. However, as beautifully demonstrated in the paper [Harris2000] found in this special issue, a program such as OTTER can provide invaluable assistance.

By way of a minute taste of which of OTTER's features proved key, note that paramodulation (an inference rule for building in equality-oriented reasoning) was heavily relied upon. (As touched on in Section 1, equality-based metaarguments are often relied upon rather than relying solely on condensed detachment.) Since Harris and Fitelson intended to produce a proof based solely on condensed detachment, a means was required to convert paramodulation proofs. In that regard, Robert Veroff and William McCune each made vital contributions.

3.10 Seeking Proofs of Lesser Distinct-Variable Complexity

For each proof $\mathbf{P}$ in hand of a given result, let $k(\mathbf{P})$ be the greatest number of distinct variables in any of its deduced steps. Among the $k(\mathbf{P})$, let k be the maximum value. If one decides to seek a proof (that is missing) of the given result such that the maximum number of distinct variables in any deduced step is strictly less than k, (similar to Section 3.6) OTTER offers precisely what is needed to commence the attack.

With the following command, the program can be instructed to purge any new conclusion with, say, more than four distinct variables.

```
assign(max_distinct_vars,4).
```

Again, as in Section 3.6, an unaided researcher could do the same. However, a program such as OTTER does not suffer fatigue or annoyance at making deductions that are promptly discarded. Neither the researcher in general nor the program can look ahead and simply avoid making such unwanted deductions.

3.11 Seeking Proofs of Lesser Level Complexity

Where (by definition) the level of the clause C is 0 when c is an input clause, and, when C is deduced, the level of C is one greater than the maximum of the levels of the clauses to which the inference rule is applied to yield C, OTTER (with the following command) can be instructed to deduce all of the clauses at level 1,2, 3, and the like and in that order.

```
set (sos_queue).
```

However, from a practical viewpoint, an unrestricted use of the given command is unwise for the size of the levels grows far too rapidly. Indeed, among the restrictions, the use of subsumption is virtually required if the program is to be given a chance at reaching the assigned goal. Nevertheless, if the proof that may be missing is one of strictly smaller level than any in hand, often the proof can be found.

3.12 Seeking Fully Automated Proofs

At one end of the spectrum is the activity of proofchecking, so vital to many areas of verification. Rather closely related is the activity of proof completion, for example, the case in which an outline of the proof is in hand or many of the steps of a known proof are in hand. Farther removed is the case in which, although proofs are in hand, no proof has been found by a reasoning program *without* reliance on a portion of one of the proofs. Indeed, if one has a reasoning program complete a proof by giving the program some or many of the steps of a known proof, then (for us) the resulting proof is not fully automated. On the other hand, if one includes resonators perhaps corresponding to the steps of a related proof or corresponding to lemmas or items considered to be of some interest in the underlying theory, and if the program then finds a proof, the game has been played fairly in the sense that the resulting proof *is* fully automated. For the class of fully automated proofs, we have in mind that one or more proofs already are in hand and, usually, at least one of them is axiomatic. Although we are not permitted to rely on a portion of a proof in hand, we are permitted to rely on the knowledge of the target of the theorem or, perhaps, on some property such as an emphasis on double negation. The task of full atomation can indeed be a challenging one, but far less difficult in general than that in which no target is even known nor is it known that an axiomatic proof has ever been seen.

A methodology now exists for finding the type of missing proof in focus in this section; see [Wos2000]. The value of such a methodology is by no means limited to this class of missing proof, as discussed in the cited paper. The corresponding research correctly suggests that, when one is able (for example) to have a reasoning program proofcheck a given result and is interested in bigger game, namely, that of finding a fully automated proof, far more than the immediate objective may result. Indeed, piquant to us, the proof that is found

via total automation is often quite unlike any that one finds in the literature, yet another bonus in some cases.

4 Successes for the Various Classes of Missing Proof

In contrast to Sections 2 and 3 in which the various classes were discussed each in a separate subsection, in this section we intermingle the classes. In fact, we sometimes give a success that is relevant to more than one class.

A rapid review of our successes immediately shows how much effort we have devoted to the class of term-avoidance proofs. That emphasis was no doubt triggered by the success a few years ago in finding a proof in infinite-valued sentential calculus in which the dependence of one of Lukasiewicz's five axioms on the other for was proved *without* requiring the use of double negation. Indeed, except for the proof of a distributive law for infinite-valued logic that Rose and Rosser call (3.44), we have always been able to find a double-negation-free proof of the theorem under consideration. The following clauses present the key aspects, for the person who wishes to attack either the dependence of the fifth axiom or to attack the problem of finding a double-negation-free proof of (3.44).

```
%  Following are the implicational axioms (MV1-MV3).
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))).
%  Following is the negation axiom MV4.
P(i(i(n(x),n(y)),i(y,x))).
%  Following is the (dependent) implicational axiom MV5.
P(i(i(i(x,y),i(y,x)),i(y,x))).
%  Following is the denial of the distributive law (3.44)
-P(i(i(n(a),n(i(i(n(b),n(c)),n(c)))),n(i(i(n(i(n(a),b)),n(i(n(a),c))),
   n(i(n(a),c)))))) | $ANS(3_44).
```

At this time (early in the year 2000), we have not attempted to prove the following metatheorem: If double negation is absent from both the hypothesis and the conclusion and if the logic under study is an interesting logic, a proof can always be found that is free of double negation. Whether the cited metatheorem is true is an open question. Further, we do not know at this time whether deep consequences for logic follow from such a metatheorem. We in fact know only that, in all but one the cases we have studied, we have succeeded in finding a double-negation-free proof under the given conditions.

Regarding the class of proofs focusing on length, again the dependence of $MV5$ is a theorem that suffices. Our proof, the following, consists of 32 applications of condensed

detachment.

A Pleasing Proof of the Dependence of MV5

```
----> UNIT CONFLICT at 9146.78 sec ----> 220818 [binary,220817.1,20.1]
   $ANS(MV_5).

Length of proof is 32.  Level of proof is 20.

---------------- PROOF ----------------

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] P(i(x,i(y,x))).
3 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
20 [] -P(i(i(i(a,b),i(b,a)),i(b,a))) | $ANS(MV_5).
26 [] -P(i(x,y)) | -P(x) | P(y).
27 [] P(i(x,i(y,x))).
28 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
29 [] P(i(i(i(x,y),y),i(i(y,x),x))).
30 [] P(i(i(n(x),n(y)),i(y,x))).
40 [hyper,1,3,3] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
42 [hyper,1,3,2] P(i(i(i(x,y),z),i(y,z))).
48 (heat=1) [hyper,26,42,30] P(i(n(x),i(x,y))).
49 (heat=1) [hyper,26,42,29] P(i(x,i(i(x,y),y))).
61 (heat=2) [hyper,26,28,48] P(i(i(i(x,y),z),i(n(x),z))).
104 [hyper,1,40,40] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
105 (heat=1) [hyper,26,104,30] P(i(i(x,y),i(i(n(z),n(y)),i(x,z)))).
112 (heat=2) [hyper,26,105,27] P(i(i(n(x),n(i(y,z))),i(z,x))).
143 [hyper,1,3,61] P(i(i(i(n(x),y),z),i(i(i(x,u),y),z))).
157 (heat=1) [hyper,26,143,30] P(i(i(i(x,y),n(z)),i(z,x))).
227 [hyper,1,104,49] P(i(i(x,i(y,z)),i(y,i(x,z)))).
234 (heat=1) [hyper,26,28,227] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
269 [hyper,1,3,112] P(i(i(i(x,y),z),i(i(n(y),n(i(u,x))),z))).
399 [hyper,1,234,104] P(i(i(x,i(y,z)),i(i(u,x),i(y,i(u,z))))).
407 (heat=1) [hyper,26,399,29] P(i(i(x,i(i(y,z),z)),i(i(z,y),i(x,y)))).
426 (heat=2) [hyper,26,407,30] P(i(i(x,y),i(i(n(x),n(i(y,x))),y))).
561 [hyper,1,399,157] P(i(i(x,i(i(y,z),n(u))),i(u,i(x,y)))).
615 [hyper,1,234,407] P(i(i(i(x,y),i(z,y)),i(i(y,x),i(z,x)))).
129815 [hyper,1,561,426] P(i(x,i(i(y,n(x)),n(y)))).
212221 [hyper,1,104,129815] P(i(i(x,i(y,n(z))),i(z,i(x,n(y))))).
213082 [hyper,1,143,(2.22)21] P(i(i(i(x,y),i(z,n(u))),i(u,i(n(x),n(z))))).
214897 [hyper,1,40,213082] P(i(i(x,y),i(z,i(n(y),n(x))))).
```

```
215622 [hyper,1,407,214897] P(i(i(i(n(x),n(y)),z),i(i(y,x),z))).
215623 [hyper,1,399,214897] P(i(i(x,i(y,z)),i(u,i(x,i(n(z),n(y)))))).
216822 [hyper,1,215623,49] P(i(x,i(y,i(n(z),n(i(y,z)))))).
217049 (heat=1) [hyper,26,216822,30] P(i(x,i(n(y),n(i(x,y))))).
218806 [hyper,1,269,217049] P(i(i(n(x),n(i(y,z))),i(n(u),n(i(i(z,x),u))))).
220019 [hyper,1,615,218806] P(i(i(n(i(i(x,y),x)),n(y)),i(n(x),n(y)))).
220033 [hyper,1,21562(2.22)0019] P(i(i(x,i(i(y,x),y)),i(n(y),n(x)))).
220471 [hyper,1,234,220033] P(i(i(i(x,y),i(y,x)),i(n(x),n(y)))).
220591 (heat=1) [hyper,26,28,220471] P(i(i(i(n(x),n(y)),z),i(i(i(x,y),
   i(y,x)),z))).
220817 (heat=2) [hyper,26,220591,30] P(i(i(i(x,y),i(y,x)),i(y,x))).

Clause (220817) contradicts clause (20), and the proof is complete.
```

In contrast, the shortest proof in the literature (known to us) is Meredith's, consisting of 37 applications. We find the result quite significant in that Meredith was known to emphasize proof elegance.

The cited double-negation-free 32-step proof (found by OTTER) that $MV5$ is dependent on the set consisting of $MV1$ through $MV4$ provides a fine example of the class of lemma-avoidance proofs. Indeed, a glance at the work of Rose and Rosser shows that they relied upon lemmas they call (2.22), (3.5), and (3.51), which seems to be the case for others attacking the given theorem. However, of these three lemmas, only (2.22) is present in the OTTER proof. If open questions are appealing, in the given context, one might pursue the question of the existence of a proof of the dependence of $MV5$ in which *none* of (2.22), (3.5), and (3.51) is present.

Perhaps we can provide some aid by briefly (as promised) touching on how we found the 32-step proof. We used ancestor subsumption, a procedure that compares path lengths to the same conclusion, preferring the strictly shorter when such is found. We used demodulation to block the use of some formula when it was weakly conjectured that such blocking might lead to a shorter proof. And, among other moves, we tried different values of the heat parameter, a parameter that governs the amount of revisiting by the program to the list known as the hot list.

With a single example from among the various successes we have obtained with OTTER, we can focus on three classes of missing proof: those where metaargument proofs were the only available type; those in which a less general proof was all that was offered; and those in which a less general lemma or result was involved. In infinite-valued sentential calculus, using implication, logical **or** can be defined as $i((x,y),y)$. With the given definition, one can prove that **or** is associative. Rather than proving the obvious equality, if the preference is to play the game solely in terms of condensed detachment, one then proves two implications. Regarding the following clauses, the first two correspond to the

two implications to be proved, and the second two correspond to generalizations of what the literature offers.

```
%  Following desired associativity lemmas
-P(i(i(i(a,i(i(b,c),c)),i(i(b,c),c)),i(i(i(i(a,b),b),c),c))) |
   $ANS(lemma_2_21a).
-P(i(i(i(i(i(a,b),b),c),c),i(i(a,i(i(b,c),c)),i(i(b,c),c)))) |
   $ANS(lemma_2_21b).
%  Following desired generalizations of associativity lemmas
-P(i(i(i(a,i(i(b,c),c)),i(i(b,c),d)),i(i(i(i(a,b),b),c),d))) |
   $ANS(lemma_gen2_21a).
-P(i(i(i(i(i(a,b),b),c),d),i(i(a,i(i(b,c),c)),i(i(b,c),d)))) |
   $ANS(lemma_gen2_21b).
```

When OTTER was given the problem of proving associativity in the nongeneral form—we were unaware that the generalization held—it proved the first clause quite easily, but the second temporarily provided substantial resistance. That resistance was easily overcome by using the steps of the proof of the first clause as resonators.

Then, as is so typical of our research, we quickly turned to the objective of finding shorter and still shorter proofs of the two implications. That search eventually produced a generalization of the second clause (the fourth clause). The other element of the more general pair offered what for a while appeared to be an impossible task to complete. However, Robert Veroff succeeded in finding a proof for the more general form of the first clause, by devising a radically different approach based on his use of *sketches*. Since the proofs of the more general theorems had been completed, obviously more general proofs (relying on more general steps part of the time), we had satisfying examples for the three missing-proof classes under discussion.

For the promised example of a published proof in which various steps (of condensed detachment) were missing and supplied by OTTER, we cite Lukasiewicz's proof of the completeness of his shortest possible single axiom for the implicational fragment of two-valued sentential calculus in which he gives but 28 steps. The object was to produce a proof that contains all 28 steps, but a proof depending solely on condensed detachment such that no step is missing. OTTER succeeded, finding a proof of length 36.

For a fine example of the finding of a missing proof in the class of those not fully automated, two-valued sentential (or propositional) calculus comes into play. That area of logic admits an axiomatization consisting of a single formula. Of the single axioms for the full two-valued sentential logic, the shortest (21 symbols in length, not counting commas and parentheses) is due to Meredith [Meredith1953]. For years, all attempts to find a means for an automated reasoning program to prove *without knowledge of an existing proof* that his formula provides a complete axiomatization failed. The effort in this regard was merited

because (we assert) triumphs in fully automated proof search can lead to a methodology that can be used to answer some deep open question or used to find an axiomatic proof where none was previously available or used in some other significant way. Our target throughout our study was the Lukasiewicz three-axiom system consisting of (what he calls) theses 1, 2, and 3.

Finally, we did develop the needed methodology, relying heavily on the use of the resonance strategy, on lemma adjunction, and on the avoidance of double-negation terms. That methodology, as will shortly be discussed, was soon put to good use in the context of another class of missing proofs. The Meredith proof consists of 41 applications of condensed detachment. Although we have been able to find a slightly different proof of length 41—his and that we found both rely on double negation, 17 steps of the type—we have never found a shorter proof and do not know whether such exists. Therefore, we offer for the interested the question focusing on the existence of a proof of length 40 or less.

Meredith's single axiom serves yet another purpose, that focusing on the class of missing proofs in the context of reducing the number of distinct variables that occur in the deduced steps. Both cited 41-step proofs rely on formulas in which seven distinct variables occur. Deepak Kapur found a 63-step proof (private communication) that Meredith implies the three-axiom Lukasiewicz system such that no deduced step relies on more than six distinct variables. When an attempt was made to find a shorter proof with the six-variable property, it succeeded, yielding a 54-step proof whose input file can be found in [Wos1999].

The cited methodology (that finally yielded the first fully automated proof of the Meredith single axiom) played the key role when we turned to the class of proofs that are missing because of being omitted. Specifically, Lukasiewicz (before Meredith) offered a single axiom for two-valued sentential calculus, one of length 23 letters, where Meredith's is of length 21. Lukasiewicz, however, did not include a proof, and (from what we know) the literature does not offer a proof—it was missing. In part to test the new methodology and in part to (we hoped) find the missing proof, we proceeded as we did when success occurred in the context of Meredith's single axiom.

Four runs and approximately 4 CPU-hours sufficed: OTTER found a 200-step proof. As predictable from what has been written in this article, the proof is free of double negation. We strongly conjecture that Lukasiewicz's proof must not have shared this intriguing property.

For an example of a proof that was missing in the context of the class concerned with formula complexity, again two-valued sentential calculus suffices, just the implicational fragment. Indeed, the Lukasiewicz proof for his (shortest possible) single axiom [Lukasiewicz1970] contains a formula of length 32 (in symbol count), whereas we have a proof whose longest deduced formula has length 24. Also of note, the Lukasiewicz proof requires the use of formulas in which eight distinct variables occur; the cited 24-step proof requires but five.

As for finding proofs of lesser level, again two-valued sentential calculus serves well: the (C,O) system [Meredith1953]. For one of his single axioms, Meredith gives a proof of length 37, relying on 8 distinct variables, and level 30; the longest formula has length 32. OTTER found a proof for the same axiom, with length 26, relying on 6 distinct variables, and level 21; its longest formula has length 24. In other words, when compared with the Meredith proof (which we give immediately), the OTTER proof is strictly less complex in all four areas we discuss in this article.

A Paradigmatic and Elegant Proof

```
Length of proof is 26.  Level of proof is 21.

---------------- PROOF ----------------

29 [] -P(i(x,y)) |  -P(x) | P(y).
32 [] P(i(i(i(i(i(x,y),i(z,O)),u),v),i(i(v,x),i(z,x)))).
33 [hyper,29,32,32] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
35 [hyper,29,32,33] P(i(i(i(x,i(O,y)),z),i(u,z))).
39 [hyper,29,35,32] P(i(x,i(i(i(O,y),z),i(u,z)))).
40 [hyper,29,39,39] P(i(i(i(O,x),y),i(z,y))).
42 [hyper,29,33,40] P(i(x,i(y,i(z,y)))).
43 [hyper,29,42,42] P(i(x,i(y,x))).
47 [hyper,29,32,43] P(i(i(i(x,i(i(i(y,z),i(u,O)),v)),y),i(u,y))).
54 [hyper,29,32,47] P(i(i(i(x,y),z),i(i(i(y,u),i(x,O)),z))).
55 [hyper,29,54,54] P(i(i(i(x,y),i(i(z,u),O)),i(i(i(u,v),i(z,O)),x))).
57 [hyper,29,32,54] P(i(i(i(i(i(i(x,O),y),i(i(z,u),O)),v),z),i(x,z))).
65 [hyper,29,54,55]
P(i(i(i(i(i(x,y),O),z),i(i(u,v),O)),i(i(i(y,w),i(x,O)),u))).
73 [hyper,29,54,57]
P(i(i(i(x,y),i(i(i(i(i(z,O),u),i(i(x,v),O)),w),O)),i(z,x))).
82 [hyper,29,32,65] P(i(i(i(i(i(x,y),i(z,O)),u),i(z,x)),i(v,i(z,x)))).
87 [hyper,29,73,65] P(i(x,i(i(i(y,i(i(x,O),z)),O),u))).
97 [hyper,29,54,87]
P(i(i(i(x,y),i(z,O)),i(i(i(u,i(i(i(z,x),O),v)),O),w))).
107 [hyper,29,82,97] P(i(x,i(i(i(y,i(i(i(z,i(u,v)),O),w)),O),u))).
114 [hyper,29,107,107] P(i(i(i(x,i(i(i(y,i(z,u)),O),v)),O),z)).
117 [hyper,29,32,114] P(i(i(x,y),i(i(i(z,i(x,u)),O),y))).
118 [hyper,29,117,117] P(i(i(i(x,i(i(y,z),u)),O),i(i(i(v,i(y,w)),O),z))).
132 [hyper,29,32,118] P(i(i(i(i(i(x,i(y,z)),O),u),v),i(i(y,u),v))).
137 [hyper,29,132,82] P(i(i(x,i(x,y)),i(z,i(x,y)))).
138 [hyper,29,132,57] P(i(i(i(x,y),x),i(z,x))).
142 [hyper,29,132,32] P(i(i(x,y),i(i(y,z),i(x,z)))).
```

```
154 [hyper,29,137,138] P(i(x,i(i(i(y,z),y),y))).
180 [hyper,29,154,154] P(i(i(i(x,y),x),x)).
198 [hyper,29,180,40] P(i(0,x)).
```

5 Milestones Reached

For decades, one of the goals of automated reasoning focused on the ability of a program to produce one proof after another when asked to do so. That milestone has been reached, at least in various areas of logic. Indeed, our submission to OTTER of one purported theorem after another in a wide variety of sentential calculi is typically soon followed by OTTER's return of a proof. Sometimes, relying on one of a set of new methodologies, a sequence of runs is required, each building on the results of the preceding. Almost never does the effort fail to find a desired proof.

A second objective of the field has been to add new proofs and new theorems to mathematics and to logic. McCune's remarkable success with the settling of the Robbins algebra question is but one of numerous such successes, successes that included results from years earlier in ternary Boolean algebra, finite semigroups, and equivalential calculus. The list continues to grow, the latest concerning the answering of two open questions posed by Epstein.

In our view, closely related to the preceding is the finding of missing proofs, which is the focus of this article. Successes in this context indeed add to mathematics and to logic, especially (in our view) when the proof that is found is axiomatic. In particular, the replacement of a proof by induction by one that relies solely on a given inference rule (such as condensed detachment) can lead by inspection to insights that in turn lead to answers to other open questions. In addition, from the viewpoint of automated reasoning, attempts to produce axiomatic proofs to fill a void in the literature can, as it has in our research, culminate in new and useful methodologies.

More distantly related to the answering of questions posed by great minds (such as that frequently asked by Tarski in the context of Robbins algebra) is the seeking of proofs satisfying some list of constraints on their structure. Obviously, as emphasized in this article (especially in Sections 2 and 4), when such proofs are not in hand, then types of missing proof can be sought. Among our successes in this arena, we cite but two at this point. The first concerns the finding repeatedly of proofs in which double-negation terms are avoided even though the literature suggests that their use is required. Indeed, although their existence might be counterintuitive, a program such as OTTER is impressively effective at finding proofs avoiding terms of the form $n(n(t))$ for some term t. The second example focuses on finding proofs of deep theorems whose proof had clearly eluded some of the best minds. One of the finer successes concerns the proving of distributivity in infinite-valued sentential

calculus.

From the global perspective, we suggest that the field of automated reasoning has made great strides in but the past five years. For evidence, we cite the examples presented in this article, as well as the material presented in the other articles offered by this special issue.

References

[Epstein1994] Epstein, R., *The Semantic Foundations of Logic: Propositional Logics,* 2nd ed., Oxford University Press, New York, 1994.

[Harris2000] Harris, K., and Fitelson, B., "Distributivity in Lw and Other Sentential Logics", *J. Automated Reasoning* (this issue), 2000.

[Hilbert1950] Hilbert, D., and Ackermann, W., *Principles of Mathematical Logic,* Chelsea, New York, 1950.

[Lukasiewicz1970] Lukasiewicz, J., *Selected Works,* edited by L. Borokowski, North Holland, Amsterdam, 1970.

[McCune1989] McCune, W., *OTTER 1.0 users' guide,* Tech. Report ANL-88/44, Argonne National Laboratory, Argonne, IL, January 1989.

[McCune1997] McCune, W., "Solution of the Robbins Problem", *J. Automated Reasoning* 19, no. 3 (1997), 263–276.

[Meredith1953] Meredith, C. A., "Single Axioms for the Systems (C,N), (C,O), and (A,N) of the Two-Valued Propositional Calculus", *J. Computing Systems* 1, no. 3 (1953), 155–164.

[Meredith1958] Meredith, C. A., "The Dependence of an Axiom of Lukasiewicz", *Trans. AMS* 87, no. 1 (1958), 54.

[Meredith1963] Meredith, C. A. and Prior, A., "Notes on the Axiomatics of the Propositional Calculus", *Notre Dame J. Formal Logic* 4, no. 3 (1963), 171–187.

[Prior1960] Prior, A., *Formal Logic,* Clarendon, Oxford, 1960.

[Rose1958] Rose, A., and Rosser, J. B., "Fragments of Many-Valued Statement Calculi", *Trans. AMS* 87 (1958), 1–53.

[Wajsberg1977] Wajsberg, M., *Logical Works,* Polish Academy of Sciences, Warsaw, 1977.

[Wos1996] Wos, L., "OTTER and the Moufang Identity Problem", *J. Automated Reasoning* 17, no. 2 (1996), 215–257.

[Wos1999] Wos, L., and Pieper, G. W., A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning. Singapore: World Scientific, 2000.

[Wos2000] Wos, L., "Conquering the Meredith Single Axiom", *J. Automated Reasoning* (this issue), 2000.