Conquering the Meredith Single Axiom*

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

May 10, 2000

Abstract

For more than three and one-half decades beginning in the early 1960s, a heavy emphasis on proof finding has been a key component of the Argonne paradigm, whose use has directly led to significant advances in automated reasoning and important contributions to mathematics and logic. The theorems that have served well range from the trivial to the deep, even including some that corresponded to open questions. Often the paradigm asks for a theorem whose proof is in hand but that cannot be obtained in a fully automated manner by the program in use. The theorem whose hypothesis consists solely of the Meredith single axiom for two-valued sentential (or propositional) calculus and whose conclusion is the Lukasiewicz three-axiom system for that area of formal logic was just such a theorem. Featured in this article is the methodology that enabled the program OTTER to find the first fully automated proof of the cited theorem, a proof with the intriguing property that none of its steps contains a term of the form $n(n(t))$ for any term t. As evidence of the power of the new methodology, the article also discusses OTTER's success in obtaining the first known proof of a theorem concerning a single axiom of Lukasiewicz.

Keywords:automated reasoning, double negation, Meredith single axiom, propositional calculus, two-valued sentential calculus

*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.

1 The Original Problem

For many years, the Meredith single axiom for two-valued sentential (or propositional) calculus had successfully resisted a fully automated proof.

```
% Following is Meredith's axiom expressed in clause notation.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
```

Although the automated reasoning program OTTER [McCune1989] was speedily able to proof check what amounts to the original 41-step proof of the Meredith single axiom, no combination of strategies and parameter settings yielded a proof that was not strongly guided by knowledge of that original proof. (Note that proof checking, especially when condensed detachment is the inference rule to be used, is far, far easier than proof finding; their natures differ sharply. Also of note, Meredith's original proof, in contrast to the cited 41-step proof, was not based strictly on condensed detachment; indeed, his 38-step proof relies in part upon substitution and in part on detachment. More generally, be warned that proof length, and proof itself, in the literature can be misleading, as the preceding illustrates.) The target, chosen by Meredith and featured in the research discussed here, was the Lukasiewicz three-axiom system.

```
% Lukasiewicz 1 2 3, expressed in clause notation.
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(n(x),x),x)).
P(i(x,i(n(x),y))).
```

This article focuses on the conquering of the axiom—on the formulation of a methodology that yields a fully automated proof that begins with the Meredith single axiom and completes with the deduction of the three Lukasiewicz axioms. The methodology relies in no way on knowledge of the proof being sought. Moreover, as required by any methodology that is to be respected, its use led to a distinct and startling success (to be presented here), namely, completion of a proof that the Lukasiewicz 23-letter axiom is sufficient for the study of all of two-valued sentential (or propositional) calculus. Before the development of the methodology featured in this article, no proof of this single axiom was known. The factors that led to the conquest and replaced the years of failure are detailed.

Also discussed in this article is the topic of shortening proofs, and the resultant algorithm I was able to formulate for making such attempts. The algorithm, as explained in Section 5, is the 1's complement to the methodology featured here.

2 A Burning Question

Especially for the individual unfamiliar with automated reasoning, a natural and even burning question immediately arises. Why spend time and effort trying to find a means for a program to prove a theorem whose proof is readily available? From a historical viewpoint, the answer rests with the continued practice of the Argonne group, namely, first identify a theorem whose proof is out of reach of our current program and then wrestle with that theorem until a means is found to bring its proof within range. From the viewpoint of automated reasoning, the general principle asserts that progress is quite likely to occur when the arsenal of weapons is augmented to permit the proof of a theorem that had resisted automation. Whether the theorem has already been proved or simply conjectured to be true is not the crucial element, although the latter case produces more excitement.

One danger always exists: The new approach may be consciously or unconsciously tailored to the theorem under consideration. To be of interest and to offer the strong possibility that an advance has occurred, such tailoring must be absent. Therefore, the supposed advance must be put to the test of applying it to similar theorems or, even better, to some not so similar. A number of the weapons now offered by a program such as OTTER can trace their birth to the study of a theorem whose proof was already in hand.

3 The Key Factors

Four factors appear to account for the success to be reported here in some detail. First, directly at the suggestion of my colleague Branden Fitelson, the heat parameter was assigned the value 8. Second, a series of lemmas (whose choice is in *no* way based on the theorem under attack) was included as targets that, when proved, were then added to the initial set of support for the next run, and the process was repeated until a proof of the desired theorem was obtained. Third, at some point, all of the proof steps of the already-proved lemmas, even those thought to be irrelevant to the task, were adjoined as resonators. Fourth—and indeed counterintuitive—the goal of finding a proof was replaced by the goal of finding a proof totally free of double negation. From what I know and from what Fitelson has found, prior to the study reported here, no such proof for the Meredith single axiom had been offered.

Before turning to the details of the methodology, I first offer a fuller treatment of each of the four factors.

3.1 Turning up the Heat

The first factor focuses on the use of the hot list strategy [Wos1999b]. My usual approach is to set the input heat parameter to a value no greater than 3. When the theorem under attack relies on a single axiom, however, a value of 8 or even greater is recommended. Indeed, as my colleague Fitelson pointed out, if you examine a proof of a theorem of the type under discussion, you often find a large number of its steps have as a hypotheses the single axiom. (He was studying the use of condensed detachment, with a heavy emphasis on the 41-step proof of the Meredith single axiom.) Moreover, you also frequently find that a sequence of such steps occurs consecutively, with the second based on the first, the third on the second, and the like. The two observations taken together suggested to him that an unusually high value assigned to the input heat parameter, coupled with the placement of the single axiom in the initial hot list, might enable OTTER to find such tightly connected sequences of steps.

For example, if a retained clause $C1$ has heat $= 1$, if the input value is 8, and if the initial hot list contains both the clause corresponding to the single axiom and that corresponding to condensed detachment, then (before leaving the scene) $C1$ would be used with condensed detachment and the single axiom to yield $C2$. Similarly, $C2$ would be used to deduce $C3$ (assuming that $C2$ was retained), and the process could continue until $C8$ was deduced and retained.

I have found that Fitelson's analysis exhibits excellent insight. Therefore, I recommend assigning a value of 8 or greater to the (input) heat parameter when the theorem under attack focuses on a single axiom.

3.2 Adding to the Initial Set of Support

The second factor concerns amending the initial set of support for a subsequent run. You might (as I did) puzzle over the value of such a move if the resonance strategy were already in use and the resonators corresponding to the set of target lemmas were already present. Indeed, on the surface, if one of the target lemmas was deduced (say, in the first run) and its resonator was already present, then it would quite quickly become the focus of attention to initiate applications of the inference rules in use. Therefore, how could the program's attack be enhanced in the so-called second run by having such a lemma included in the input set of support? In particular, the two approaches would seem to be essentially equivalent.

As Fitelson and I hazarded in a phone conversation, here is the probable difference (by example). If the target lemma is deduced and retained (rather than being present at the beginning of the run) at, say, clause 1000 and used to deduce a complex clause that is retained and given, say, the number 20,000, then the clause numbered 20,000 might never be chosen as the focus of attention for inference-rule initiation. Even with the presence of the ratio strategy and a value of 4 assigned to the pick_given_ratio, OTTER might never (or almost never) choose clause 20,000 as the focus of attention.

On the other hand, in a run in which the clause corresponding to the target lemma (numbered 1000 in the preceding comments) is included in the initial set of support, the clause that would have been numbered 20,000 might now be numbered 500 or less, and hence will be chosen as the focus of attention far sooner. In other words, the search space has been perturbed, and perturbed in a most profitable fashion.

The discussion also sheds some light on the difference between lemma inclusion and lemma adjunction. Of particular interest is the contrast between the way a researcher works and the way a program works in the context of inclusion versus adjunction. A person will use the lemma when convenient, regardless of its so-called number in memory; a program might never use it because of its being assigned a large conclusion number (see [Wos1999c] for further discussion of this point).

You now might be ready for a new strategy (not offered by any program known to me), one somewhat in the spirit of learning. In the context of OTTER, it works this way. Place on the passive list the negations of various target lemmas that might merit proof. As each is proved, immediately add the (positive form of the) lemma to the remainder of the initial set of support. Of course, if you have used set(input_sos_first), then most likely all of those clauses will have been already chosen as the focus of attention. In that case, take the (positive) form of the lemma (where its negation is on the passive list) and immediately choose it as the focus of attention.

Related to the given strategy is another strategy. The crux is the immediate adjunction to the hot list of such a proved lemma. Also related is the strategy that immediately adds to the resonators the correspondent of the just-proved lemma. In such an event, the program might benefit from then changing all of the weights of the clauses remaining on the set of support list, changing the weights commensurate with the new resonator. (Neither of the two just-cited strategies is offered by any program with which I am familiar.)

As for the now key question concerning where to obtain the target lemmas for the passive list, I suggest any book, any paper, or any research of which you know that concerns a theorem related to that under study. For example, part of my success with the Meredith theorem resulted from using as targets 68 theses (proved by Lukasiewicz and numbered 4 through 71) in his study of his three-axiom system for two-valued sentential (or propositional) calculus.

3.3 Adjoining Resonators

The third factor leading to the successes reported in this special issue concerns adding as resonators all proof steps of proved target lemmas. The notion is that, if such steps were useful in proving a target lemma, then a step of the same pattern might be of use in reaching the main objective. To emphasize the point, I am not suggesting that the actual proof steps might prove useful; rather, I am talking about the symbol pattern in the resonator sense, where all variables are treated as indistinguishable.

3.4 Avoiding Double Negation

The last of the four factors is, without question, the most startling and therefore the one deserving the most comment. When you are asked to find a proof, but also requested to satisfy additional constraints, intuition strongly and correctly suggests that in general the task becomes more difficult. For example, rather than seeking *any* proof, you are asked to seek one in forty or fewer applications of some given inference rule. For a second example, you are asked to find a proof that avoids the use of some specified lemma. For a third example (directly pertinent to this article), you are asked to find a proof in which no term in any proof step has the form $n(n(t))$ for some term t. A proof within an added given constraint may not even exist. Therefore, seeking a proof that avoids the use of double negation would naturally seem harder than completing a proof not satisfying such a constraint. Indeed, a glance at the pertinent literature shows that the use of double negation abounds.

But, in fact, adding such a constraint appears to have helped to find a proof. I conjecture that the space of conclusions for the automated reasoning program OTTER to browse in has been markedly reduced in size. Perhaps the presence of conclusions containing $n(n(t))$ terms is not so loathsome in itself, but (perhaps) such conclusions spawn an intractable space of retainable conclusions. In other words, perhaps the density of good information within the total information deduced and retained is far greater even though the added constraint clearly guarantees that the number of acceptable proofs is sharply reduced.

This conjecture is in part based on my repeated attempts (with no success) to obtain a fully automated proof of any type (for the Meredith single axiom), before adding the constraint of avoiding double negation. As a result, I suggest the following aphorism, at least if you are using a reasoning program such as OTTER.

- If you are meeting apparently *unconquerable* resistance, then replace the question by a *harder* question, or replace the goal by one that appears to be *more unreachable*!

You might wonder why I chose to focus specifically on term avoidance. The main impetus was my success several years ago in finding a proof in many-valued sentential calculus with such a constraint (Chapter 11 of [Wos1996],) a proof conjectured to be nonexistent. Since then, when studying some problem in a logic calculus, I frequently have attempted to find a proof free of double negation. And so I made such an attempt in my most recent study of the Meredith single axiom (completing with the deduction of the three-axiom system of Lukasiewicz).

4 A Methodology Is Born

The use of a methodology based on the four cited factors did, in fact, yield a fully automated proof that the Meredith single axiom implies each of the three Lukasiewicz axioms, $L1$, $L2$, and $L3$. Only four runs (experiments) were required.

For the first experiment, as is typical when I conduct a sequence of tightly coupled experiments, intuition was the main source for the values assigned to the various parameters. My experiences with OTTER have taught me about tendencies, but I can offer no firm rules for accurately making choices guaranteed to bring reward. (Some guidelines for the use of OTTER are provided in Chapter 9 of [Wos1999a], and some tendencies are discussed in [Wos1998].) In general, a first experiment provides useful information, both positive and negative, regarding how to proceed in later experiments. Here are values I assigned for Experiment 1, with comments to follow; the full input file is presented in the Appendix. (Far fewer details will be given for the second through the fourth experiments.)

```
assign(max_weight,32).
assign(change_limit_after,2000).
assign(new_max_weight,24).
assign(max_proofs,-1).
assign(max_distinct_vars,6).
assign(pick_given_ratio,4).
assign(max_mem,480000).
assign(report,3600).
assign(heat,8).
%  assign(dynamic_heat_weight,0).
```

With virtually no exception, I find it wise to place an upper bound on the complexity of newly retained conclusions (whether measured in terms of symbol count or based on included weight templates); without such a bound, the program may quickly drown. In part influenced by the fact that the Meredith axiom is expressed with 22 symbols (counting its predicate symbol) and in part by experiments in two-valued sentential calculus that suggested room must be given for formulas longer than the hypothesis, I chose the value 32 for the value to assign to max_weight (the upper bound just cited). Especially because but one axiom is present from which to reason, a fair amount of room must be given for retaining new conclusions. However, being almost certain that memory would be consumed too quickly if no action was taken, I reduced the max_weight to 24 after 2000 clauses were chosen to initiate applications of condensed detachment, the inference rule in use. As shown, OTTER was assigned the value 480000 for max_mem, enabling it to rely on 480 megabytes for its search.

To permit the program to occasionally focus on a retained conclusion of much complexity (possibly 32), I assigned the value 4 to the pick_given_ratio. As it turned out, twenty-one of the conclusions chosen to initiate inference-rule application had a weight (complexity) of 32. By assigning the value 4, the program was instructed to choose 4 clauses by complexity, then 1 by first-come first-served, then 4, then 1, and the like.

To complete as many proofs as memory would allow, I assigned the value -1 (infinity) to the max_proofs parameter. To follow the activity of OTTER, I asked for a statistical report every 3600 CPU-seconds.

Regarding the assignment of the value 6 to max_distinct_vars—preventing the program from retaining any new conclusion if it relied upon 7 or more distinct variables—I regret to note that prior knowledge was used. However, such reliance was not necessary. Indeed, if the value 4 is assigned, OTTER quickly informs the researcher that nothing of value will result. If the value 5 is assigned, again nothing of consequence occurs; but two clauses are deduced, and none are retained.

The last assignment (influenced by my joint attack with Fitelson on diverse questions from mathematics and logic) was the value 8 to the heat parameter. Again, prior knowledge came into play. Specifically, he noted the occurrence of sequences of steps (in proofs in systems relying on a single axiom) all of which had as a parent the single axiom and, as the other parent, the preceding step. Access to such sequences is made far, far easier by use of the hot list strategy with a moderately high value assigned to heat and with the single axiom placed in the (initial) hot list. As it turned out (in Experiment 1), only three clauses with heat=8 were chosen as inference-rule initiator, two with heat=7, two with heat=6, and six with heat=5.

Were any of those clauses crucial for proof completion? A glance at the corresponding output file showed that one of the clauses with heat=6 was used in six different proofs; therefore, although the value 8 often does serve well, the value 6 would have sufficed at this point, which can only be known after the fact. Evidence was thus present for the value of assigning a (for me) unusually high value to heat; seldom do I rely on a value greater than 3. Indeed, *the first of the four key factors came into play.*

As for the "set" options, hyperresolution was set because of reliance on condensed detachment. The option order_history was set to permit the user to see which items were paired with the major premiss and which with the minor premiss in the three-literal clause for condensed detachment. Finally, out of habit, input_sos_first was set to cause OTTER to choose for inference-rule initiators all clauses in the input set of support before choosing any deduced clause. (Of course, since only one clause was included in the initial set of support, that for Meredith's axiom, the command was unneeded.)

By assigning the value 8 to the heat parameter and by placing the Meredith axiom with the clause for condensed detachment in the input hot list, I provided a means for the program to emphasize the role of the Meredith axiom. OTTER was thus encouraged to seek a proof that shared with Meredith's proof (found in the Appendix) what might be termed the recursive emphasis on the single axiom. However, previous failures with attempts to find (in a fully automated fashion) a proof more or less emulating that of Meredith, coupled with previous successes with other theorems when term-avoidance was imposed, strongly suggested that the program should be prevented from relying on double negation (which Meredith's proof relies upon). Of the forty-one steps of what amounts to his proof, seventeen rely on the use of a term of the form $n(n(t))$ for some term t. The following set of demodulators was used to achieve the cited term avoidance.

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

The fourth of the four factors was thus brought into play.

With the program's reasoning restricted by blocking the use of double negation, I turned next to directing its reasoning. (Of course, the set of support strategy was being used to restrict the program's reasoning, but, because of the focus on a single axiom and on condensed detachment, in the obvious sense, its use added little to the program's power.) Resonance was the choice for directing the reasoning. But which resonators were appealing, resonators that must be chosen without knowledge of Meredith's original proof? I chose the Lukasiewicz's sixty-eight theses (which he numbered 4 through 71). A resonator corresponding to each of the sixty-eight was placed in the pick_and_purge weight list with an assignment of the value 2. The only other resonator that was included was that corresponding to the Meredith axiom, assigned a value of 1. As evidence that the deck was not being stacked, I note that only three of the sixty-eight theses are among the forty-one steps of the Meredith proof.

In the context of eventual lemma enrichment, the same sixty-eight theses (as well as a few other lemmas), each negated, were placed in the passive list with the intention of adjoining in a later run any that were proved. Although far less important, the program's success in proving such theses was also used by me as a sign of progress; indeed, I often include negations of various lemmas in the passive list to evaluate and monitor the program's attack. *The second of the four cited factors was thus in evidence.*

Experiment 1 was encouraging, proving sixteen lemmas. Of the three Lukasiewicz lemmas of which the target consists, $L3$ was proved. Regarding members of the other known axiom systems for two-valued sentential calculus whose negations were placed in the usable list, theses 18 and 19 were also proved. Something else also occurred that I had not anticipated: some lemmas were proved more than once. Indeed, in this first experiment, thesis 10 was proved in four different ways; four different conclusions were drawn such that each provided unit conflict with the negation of thesis 10.

The stage was thus set for the second experiment, differing from the first in that the max_weight was increased from 32 to 36, the max_distinct_vars was increased from 6 to 7, and the lemmas proved in Experiment 1 were adjoined to the initial set of support for Experiment 2.

Experiment 2 was even more encouraging, proving fifteen more lemmas including $L2$ and (regarding members of the included known axiom systems) theses 30, 37, 49, and 54, as well as proving $MV5$ (which is one of the five axioms Lukasiewicz originally offered for a weaker area of logic, many-valued sentential calculus). Similar to Experiment 1—and even more to my surprise—$MV5$ was proved twice, each proof terminating in a conclusion more general than $MV5$. Although the target was still the Lukasiewicz three-axiom system, OTTER and the methodology were getting quite close to proving other axiom systems, any of which would have been acceptable.

In Experiment 3, *the third of the cited four factors now came into play,* namely, the addition of 242 resonators (removing duplicates) corresponding to proof steps of lemmas proved in Experiments 1 and 2. Each was assigned the value 1 to instruct OTTER to give any matching and retained new conclusion very high preference for initiating inference-rule application. Also, the initial set of support used for Experiment 2 was extended by adjoining fifteen lemmas. This third experiment proved $L1$, and it also noted that the entire three-axiom system of Lukasiewicz had been proved. As evidence of how hard OTTER had to work, the proof of $L1$ completed with the retention of clause 968418.

Elation was certainly my reaction, coupled with being startled because the 480 megabytes of memory allotted for the experiment were exhausted almost immediately after $L1$ was deduced. Of course, the task was not finished; indeed, a stand-alone proof had not yet been produced. In particular, what was available at the completion of Experiment 3 was a proof resting on the use of lemmas proved in that experiment and on lemmas proved in each of the two preceding experiments. Also, in the spirit of recursion, some of the used lemmas from Experiment 2 rested on lemmas proved in Experiment 1. I strongly preferred to find a means to produce the stand-alone proof *without* untangling the cited dependencies.

I thus tried Experiment 4 in which I commented out all of the items in the initial set of support of Experiment 3 other than the Meredith single axiom. That move was required to enable OTTER to attempt to find a stand-alone proof. I also added two sets of resonators, with the intention of aiding the program, based on my success in Experiment 3. The first set (of fifty-eight) corresponded to the claimed proof of the target axiom system; the second set corresponded to proof steps (sorted) from Experiment 4 not already used. The value 0 was assigned to the members of the first set, and the value 1 was assigned to the members of the second set.

The hard work paid off: Experiment 4 yielded the sought-after proof of the Lukasiewicz three-axiom system in less than 90 CPU-seconds, a proof of length 160 (applications of condensed detachment) and level 74. That proof was followed by proofs of the Hilbert system, the Church system, the alternate system of Lukasiewicz, and my axiom system.

Virtually required at this point is a pause to consider the claim that the 160-step proof was obtained in a fully automated manner. Stated differently, how much did I cheat? First, of the 160 steps, only 5 are among the theses 4 through 71, which were the key resonators for the first experiment; the 41-step Lukasiewicz proof contains but three of the cited 68 theses. This evidence nicely addresses the question of cheating. Nevertheless, there remains the question of how closely tied the 160-step proof is to Meredith's proof. Of the 24 steps of the Meredith proof that are free of double negation, 21 are among the 160 steps of the proof yielded by the methodology. A natural question immediately arises concerning how crucial are the 21 steps, viewed as lemmas, a question that perhaps is answered in Section 5.

The primary objective had been reached—a fully automated proof that Meredith's axiom is sufficient for the study of two-valued sentential (or propositional) calculus. Nevertheless, two additional objectives remained. First and foremost, the newly formulated methodology demanded to be tested on another theorem, in part to demonstrate that the dice were not loaded to take advantage of the case just presented, and in part to gain some insight into the power of the methodology. Second, as I so often do, I wished to make a serious attempt to find a proof substantially shorter than the 160-step proof, still avoiding the use of double negation.

5 An Algorithm for Finding Shorter Proofs

The focus in this section is on the second of the two cited objectives (proof shortening), delaying the best for last. The forces that lead to seeking a proof shorter than that in hand are many and diverse, ranging from the intellectual to the practical. At one end of the spectrum, the curiosity of researchers is at stake. Can a proof be found that avoids the use of one or more specified lemmas? All being equal, the shorter the proof, the more likely that unwanted lemmas can be dispensed with. Can it be proved that the shortest possible proof has been produced? At the other end of the spectrum, in contrast to simple curiosity, economics comes into play. For example, if the shorter proof is completed in the context of circuit design, then a correspondingly more efficient circuit may have been constructed.

OTTER already has an approach to finding shorter proofs, namely, the use of ancestor subsumption, which works in the following manner. First note that the clause A *properly subsumes* the clause B if and only if A subsumes B but B does not subsume A. Second note that the *derivation length* of the clause A is the number of distinct steps in the deduction of A, not including those that are among the input; thus the derivation length is equal to the number of applications of the inference rule or rules used to deduce A. Finally, by definition, the clause A *ancestor-subsumes* the clause B if and only if (1) A properly subsumes B or (2) A and B are alphabetic variants and the derivation length of A is less than or equal to that of B. If ancestor subsumption is in use and if the length of a derivation (of a clause) is strictly less than that of an already-retained copy of that clause, then, if back subsumption is in use (which is strongly recommended), the copy is retained in the set of support list, with the original copy of the clause purged from the database (used for reasoning) and "hidden" to be recalled in the event it occurs in a completed proof.

While ancestor subsumption is often effective in finding shorter proofs, researchers quite naturally prefer an algorithm. The algorithm I developed (first discussed in print in [Wos1999a] has exhibited unanticipated power. Although its use does not guarantee the return of a strictly shorter proof, at least the first few applications typically are quite rewarding. (Interesting, although perhaps not obvious, is the fact that the use of ancestor subsumption can result in a program completing a longer proof than it does in its absence, as occurred in one of my experiments with the Meredith single axiom. Briefly, the explanation rests with the possible exclusion of intermediate steps that, if present, would have been repeatedly used later.) The algorithm works in the following way.

The algorithm requires access to a proof. Let the proof length be k, the number of deduced steps (not counting those in the input). For the first step of the algorithm, the program is asked to make k runs. In each run, one of the k deduced proof steps is prevented from being used. Although weighting suffices, I prefer to rely upon demodulation, demodulating the unwanted conclusion to junk. (The use of weighting can block items that are similar, are cousins, where all variables are treated as indistinguishable; the use of demodulation will not have this effect and, instead, blocks items that are subsumed by the item being blocked.) In the second step, all of the proofs completed in the k runs are examined to see whether a shorter proof has been completed. A run may yield more than one proof, for example, if ancestor subsumption is used. Let j be the length of the shortest proof found with the k runs. If j is greater than or equal to k, cease the attack. If such is the case, because the algorithm is usually applied repeatedly, a shorter proof may have already been found, shorter than the one that initiated the investigation. If j is strictly less than k, then apply the third step. For the third step, the focus in on a proof of length j, with the objective of finding a still shorter proof. Two choices exist: (1) the program can be asked to avoid the use of each of the j steps of the proof in focus, and (2) the program can be asked to avoid the use of each of the k steps of the initiating proof.

A variation on the algorithm that I have found quite effective is to ask the program to attempt to complete a proof when all of the steps that individually produced a proof of length j, with j strictly less than k, are simultaneously prevented from being used. An alternative stop condition has also proved of use. Specifically, if say at least ten different steps when blocked yield a proof of length j with j strictly less than k, terminate the attack.

By using this algorithm (which began by focusing on the 160-step proof yielded by applying the methodology presented earlier) coupled with frequent use of ancestor subsumption, I was able to find a far more attractive proof for the Meredith single axiom, one free of double negation. The length of that proof is 98.

However, earlier experiments had yielded a 90-step proof—which I shall call proof 1—a proof found with the algorithm and with other techniques, beginning with a 181-step proof (one whose origin may have been the methodology, but perhaps not). Therefore, I pressed on, but with no progress. (Sometimes, in a later run, I include more than one new demodulator, choosing to add those that indicate the use of each would produce a proof of length j with j strictly less than k.)

To seek a breakthrough, I chose to block an entire set of formulas, all of those matching one of the resonators present in the 98-step proof. A few experiments sufficed, resulting in the finding of another 90-step proof—which I shall call proof 2. But again a stone wall was encountered, which led me to one more move.

After the set of resonators corresponding to proof 2, I included as resonators the steps corresponding to proof 1. I was not hopeful, suspecting that proof 2 would simply be reproduced. Fortunately, my conjecture was in error: A 76-step proof of level 48 was found instead; see the Appendix. The most likely explanation for this sharp advance was that clauses that would otherwise have been discarded because of being assigned a weight strictly greater than 2 (the assigned max_weight) were retained and used to initiate inference-rule application. Their use almost certainly was intermingled with the use of clauses matching one of the 90 steps of proof 2. In other words, the program was given more latitude regarding this one aspect, which in turn resulted in a severe perturbation of the space of conclusions being examined.

Piquant, however, is the fact that the methodology for finding the *first* proof in general relies on giving the program more and more latitude, in contrast to giving the program in general less latitude when the objective is to find a far shorter proof based on the first proof. Adding elements to the set of support for a succeeding run (as is part of the methodology for finding a first proof) is in a sense the 1's complement to blocking the use of items by means of demodulation when seeking to improve upon an existing proof.

In the context of finding short proofs, a sharp contrast exists between blocking the use of an unwanted step with demodulation and using a resonator for that purpose. The former blocks the use of the item being demodulated to junk and also all instances of it, but has no effect on items that are merely similar in functional shape. The latter blocks the formula or equation in the corresponding resonator, has no effect on proper instances of it, but prevents the use of formulas or equations that are similar in functional shape (that match the item in the resonator sense). In other words, demodulation restricts the program's reasoning by blocking the use of one set of conclusions, whereas resonance blocks the use of a quite different set of conclusions. When a given demodulator is shown to prevent the use of two conclusions, a subsumption relation exists between them. When a resonator does so, no subsumption relation exists between the two items. (To complete this aspect of the discussion, if none of the related subsumption options are exercised, use of a *hint* [Veroff1996] focuses on a formula rather than on a set of formulas as resonance does.)

Now, I turn to data relevant to the observations and the question raised near the end of Section 4. First, the 76-step proof relies on but four of the 68 theses (4 through 71) used to find the 160-step proof. Therefore, are the 68 theses needed at all, or, instead, is their presence the key to focusing on crucial steps? Second—and fascinating to me because of what it seems to imply about Meredith—of the 76 steps, 23 are present in the Meredith 41-step proof, 23 of its 24 that do not rely on double negation. Are those 24 steps, at least almost all of them, so crucial to finding a proof, especially in the context of finding a "short" proof? Therefore, to find a proof shorter than length 76, does it mean that the cited 23 (or almost all of them) must be present? Answers to questions of this type are pertinent to the larger questions: Can a proof of minimal length be found, and what is that length? Third, regarding the two 90-step proofs, perhaps the fact that they differ by 38 steps played a key role in breaking through to a 76-step proof. Fourth, only four of the 76 steps require the use of seven distinct variables. In view of the fact that this proof is free of double negation, does it contain the keys to completing a double-negation-free proof none of whose formulas requires the use of more than six distinct variables?

6 Validating the Methodology

To return to the first objective, all agree that a methodology requires testing and evaluating. In this section, the focus is on one of the finer tests that could be applied. The test removes all doubt that some form of dice-loading occurred in the conquest of the Meredith single axiom, in turn removing all doubt that prior knowledge, even if implicit or hidden from the researcher, is required.

Specifically, to test the methodology, a theorem other than that concerned with Meredith's single axiom was required. Even better, if success were to occur, would be a *purported* theorem or a theorem whose proof was unavailable, absent entirely from the literature and from the researcher's knowledge. The target for testing was (in effect) supplied by my colleague Fitelson roughly five months before the methodology was formulated. That target was a 23-letter single axiom offered by Lukasiewicz for two-valued sentential (or propositional) calculus, the following.

```
% Following is Lukasiewicz's 23-letter single axiom.
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
```

I was told that no proof was given and that (apparently) Meredith knew of none; but presumably Lukasiewicz did, and so the game was on. A condensed detachment proof must exist, if only it could be discovered. What would be a reasonable choice for establishing that the given axiom suffices? I chose the three-axiom system (cited earlier in Section 1) of Lukasiewicz.

With but four runs, as if already scripted, the methodology prevailed. Indeed, in the third experiment, OTTER completed an 82-step proof of level 26 of the three-axiom Lukasiewicz system. However, as with Meredith—since lemmas proved in preceding runs were relied upon—there remained the task of producing a proof dependent solely on the Lukasiewicz 23-letter single axiom. Adhering to the methodology, I then ran a fourth experiment and obtained the desired proof, one of length 200 and level 68. As a point of interest (and perhaps suggestive of future research), the 200-step proof relies on but eight of the 68 theses 4-71, and seven of the 200 steps require the use of six distinct variables. If max_distinct_vars is assigned the value 5, no new conclusions are produced, correctly implying that 6 is the minimum.

The unavailability in the literature of any proof for the Lukasiewicz 23-letter single axiom strongly suggests that OTTER's success is indeed singular. I do view the result as startling and as satisfying evidence that the methodology provides a powerful aid for answering open questions and for finding missing proofs. Also of note is the absence of double negation in the proof.

As might be expected, I next sought a far, far shorter proof. I turned to the coupling of ancestor subsumption and the algorithm of Section 4. The result was the completion of a 70-step proof of level 51. Three of its steps require the use of six distinct variables; of the 70 steps, 44 are present in the initiating 200-step proof; eight steps are among theses 4-71.

When the information about the 200-step proof and the 70-step proof was conveyed to a colleague of mine, she asked the following interesting question. Should the 70-step proof be considered a proof with far less depth when compared with the 200-step proof? Quite the contrary; the 70-step proof most likely would have remained hidden for decades were it not for access to the 200-step proof. Most likely, from the viewpoint of CPU time, an attempt to find the 70-step proof without knowledge of the longer proof would have occupied a very fast computer for years—if a short proof ever would have been completed. An unassisted researcher sometimes approaches a problem in a manner quite similar to that taken by a researcher assisted by an automated reasoning program.

7 Future Research

Research that succeeds can produce a type of excitement and exhilaration that defies description and that is difficult to equal. Research can also produce much frustration and even be addictive. For example, I myself find tantalizing, and even maddening, the still-unsuccessful search for a proof shorter than Meredith's 41-step proof of his single axiom. But—as was the case with the Lukasiewicz 23-letter single axiom—when a theorem is announced without proof and research yields that proof, little doubt remains concerning the balance of pain versus pleasure.

In this concluding section, I present research suggestions and observations. Some of the suggestions may aid a person who has (wisely) chosen to team with an automated reasoning program such as OTTER to find a proof that resists completion or find a shorter proof than that in hand. Some, focusing on intriguing problems, may instead provide new areas of research. Of a more general nature, the observations may stimulate research by providing insight into the contrast between the attack of a reasoning program and that of a researcher. The observations are paraphrasings from an e-mail note received from my colleague Fitelson.

My first suggestion for the researcher seeking a more elegant (shorter) proof than offered by the literature is to beware of the cited proof length. Indeed, for the following reasons, the cited length may be misleading. First, logicians (such as Lukasiewicz and Meredith) often do not apply a most general unifier, being content instead with less generality and the use of instantiation. Care, therefore, must be exercised in simply accepting quoted proof length. Second, sometimes the implication that condensed detachment is the only rule of inference in use is not accurate. Third, sometimes researchers rely on metatheorems and identities not part of the object-level theory, such as the equivalence of $n(n(t))$ and t. Such moves are (apparently) designed to avoid examination of deep levels within the tree of possible conclusions. Such moves also reduce the complexity of the expressions that must be considered; indeed, proofs often become more readable. Chess masters often rely on similar so-called tricks. In contrast, a program such as OTTER does not shy away from examining deep levels or focusing on complex expressions.

Nevertheless, to succeed, an automated reasoning program also requires moves that sharply reduce the size of the search space; strategies are an excellent example. Blocking the retention of formulas in which double negation is present is such a move—although it seems to be counterintuitive, one that an unaided researcher would not make. Restricting the number of distinct variables is yet another counterintuitive move, as is turning up the heat—but these actions often are surprisingly effective.

Once the target is established, experimentation is essential—whether the maddening hunt is for a new data structure to sharply increase program performance, a new strategy to restrict or direct reasoning in general, a new methodology, or a new proof. Admittedly, experimentation can be frustrating, and progress slow and difficult to measure. However, I suggest that the researcher who finds a new proof—even if its length is disappointing—carefully examine the result. Indeed, examination of the new proof can produce questions of the following type, questions whose answers can in turn lead to further advances. If the new proof is shorter than that of the original, are there steps from the original proof that are used in a different way to cover for the steps that have been removed? Is there a chain of inferences in the original proof that is absent in the new proof? Are there steps in both proofs that are essential for *any* proof, are indispensable lemmas? In the case in which a single axiom is being studied with condensed detachment, the first condensed detachment must focus on two copies of the axiom and, therefore, the result must be present in all proofs of all theorems that can be proved. A glance at this first step establishes a lower bound on the number of distinct variables as well as a lower bound on the length of formulas appearing in a proof.

And now, for the curious who are ready eager for a challenge, I offer several research problems. The first involves the Meredith single axiom. Like Meredith's 41-step proof, my 160-step proof, both 90-step proofs, the 181-step proof, and the 76-step proof all require the use of formulas relying on seven distinct variables. Formulas in seven distinct variables are not required to derive the Lukasiewicz three-axiom system from the Meredith single axiom. Indeed, by extending the research of my colleague Deepak Kapur (who found a proof of length 63 requiring no more than six distinct variables in any formula), I was able to find a 54-step proof with that property, and still later with a different approach I found a 51-step proof of the same type. However, the six-variable proofs known to me all rely on the use of double negation. Whether there exists a proof requiring no more than six variables but that is free of double negation is at this time (in mid-1999) an open question.

A quite different research question focuses on the application of the algorithm presented, coupled with the use of ancestor subsumption, to yield a proof of length strictly less than 76 (beginning with the 160-step proof). Also of interest is the application of the methodology to the case in which the Lukasiewicz three-axiom system (or that of Church, for example) is taken as the starting point with the objective being the Meredith single axiom or the Lukasiewicz 23-letter single axiom.

Turning to many-valued sentential calculus, one may apply the methodology featured here to find proofs of key theorems. Examples include the fifty lemmas found in Rose and Rosser [Rose1958] and theorems concerned with associativity and distributivity and possible generalizations of these laws.

Also merited is research that leads to a generalization of the methodology (central to this article) to problems in which equality plays a key role.

Welcome are theorems coupled with a proof with the objective of finding a shorter proof. Finally, of paramount interest is a proof depending solely on condensed detachment, of length less than or equal to 40, and showing that the Meredith single axiom implies the three-axiom system of Lukasiewicz.

Appendix

Input File for Experiment 1

```
% Trying for an automated proof of Meredith's single axiom.
set(hyper_res).
assign(max_weight,32).
assign(change_limit_after,2000).
assign(new_max_weight,24).
assign(max_proofs,-1).
clear(print_kept).
% set(process_input).
% set(ancestor_subsume).
clear(back_sub).
clear(print_back_sub).
assign(max_distinct_vars,6).
assign(pick_given_ratio,4).
assign(max_mem,480000).
assign(report,3600).
set(order_history).
set(input_sos_first).
assign(heat,8).
% assign(dynamic_heat_weight,0).

weight_list(pick_and_purge).
% Following are theses 4 through 71.
weight(P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))),2).
weight(P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))),2).
weight(P(i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))),2).
weight(P(i(i(x,i(i(y,z),u)),i(i(y,v),i(x,i(i(v,z),u))))),2).
weight(P(i(i(x,y),i(i(z,x),i(i(y,u),i(z,u))))),2).
weight(P(i(i(i(n(x),y),z),i(x,z))),2).
weight(P(i(x,i(i(i(n(x),x),x),i(i(y,x),x)))),2).
weight(P(i(i(x,i(i(n(y),y),y)),i(i(n(y),y),y))),2).
weight(P(i(x,i(i(n(y),y),y))),2).
weight(P(i(i(n(x),y),i(z,i(i(y,x),x)))),2).
weight(P(i(i(i(x,i(i(y,z),z)),u),i(i(n(z),y),u))),2).
weight(P(i(i(n(x),y),i(i(y,x),x))),2).
weight(P(i(x,x)),2).
weight(P(i(x,i(i(y,x),x))),2).
weight(P(i(x,i(y,x))),2).
weight(P(i(i(i(x,y),z),i(y,z))),2).
weight(P(i(x,i(i(x,y),y))),2).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),2).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).
weight(P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))),2).
weight(P(i(i(i(x,y),x),x)),2).
weight(P(i(i(i(x,y),z),i(i(x,u),i(i(u,y),z)))),2).
weight(P(i(i(i(x,y),z),i(i(z,x),x))),2).
weight(P(i(i(i(x,y),y),i(i(y,x),x))),2).
weight(P(i(i(i(i(x,y),y),z),i(i(i(y,u),x),z))),2).
```

```
weight(P(i(i(i(x,y),z),i(i(x,z),z))),2).
weight(P(i(i(x,i(x,y)),i(x,y))),2).
weight(P(i(i(x,y),i(i(i(x,z),u),i(i(y,u),u)))),2).
weight(P(i(i(i(x,y),z),i(i(x,u),i(i(u,z),z)))),2).
weight(P(i(i(x,y),i(i(y,i(z,i(x,u))),i(z,i(x,u))))),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(y,z)),i(i(x,y),i(x,z)))),2).
weight(P(i(n(x),i(x,y))),2).
weight(P(i(i(i(x,y),z),i(n(x),z))),2).
weight(P(i(i(x,n(x)),n(x))),2).
weight(P(i(n(n(x)),x)),2).
weight(P(i(x,n(n(x)))),2).
weight(P(i(i(x,y),i(n(n(x)),y))),2).
weight(P(i(i(i(n(n(x)),y),z),i(i(x,y),z))),2).
weight(P(i(i(x,y),i(i(y,n(x)),n(x)))),2).
weight(P(i(i(x,i(y,n(z))),i(i(z,y),i(x,n(z))))),2).
weight(P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))),2).
weight(P(i(i(x,y),i(n(y),n(x)))),2).
weight(P(i(i(x,n(y)),i(y,n(x)))),2).
weight(P(i(i(n(x),y),i(n(y),x))),2).
weight(P(i(i(n(x),n(y)),i(y,x))),2).
weight(P(i(i(i(n(x),y),z),i(i(n(y),x),z))),2).
weight(P(i(i(x,i(y,z)),i(x,i(n(z),n(y))))),2).
weight(P(i(i(x,i(y,n(z))),i(x,i(z,n(y))))),2).
weight(P(i(i(n(x),y),i(i(x,y),y))),2).
weight(P(i(i(x,y),i(i(n(x),y),y))),2).
weight(P(i(i(x,y),i(i(x,n(y)),n(x)))),2).
weight(P(i(i(i(i(x,y),y),z),i(i(n(x),y),z))),2).
weight(P(i(i(n(x),y),i(i(x,z),i(i(z,y),y)))),2).
weight(P(i(i(i(i(x,y),i(i(y,z),z)),u),i(i(n(x),z),u))),2).
weight(P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))),2).
weight(P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))),2).
weight(P(i(i(x,y),i(i(z,y),i(i(n(x),z),y)))),2).
weight(P(i(i(n(n(x)),y),i(x,y))),2).
weight(P(i(x,i(y,y))),2).
weight(P(i(n(i(x,x)),y)),2).
weight(P(i(i(n(x),n(i(y,y))),x)),2).
weight(P(i(n(i(x,y)),x)),2).
weight(P(i(n(i(x,y)),n(y))),2).
weight(P(i(n(i(x,n(y))),y)),2).
weight(P(i(x,i(n(y),n(i(x,y))))),2).
weight(P(i(x,i(y,n(i(x,n(y)))))),2).
weight(P(n(i(i(x,x),n(i(y,y))))),2).
% Following is Meredith's axiom.
weight(P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))),1).
% % Following is recursive tail strategy.
% weight(i($(1),$(2)),1).
end_of_list.


list(usable).
% condensed detachment
-P(i(x,y)) | -P(x) | P(y).
% The following disjunctions are known axiom systems.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(n(n(p)),p)) | -P(i(p,n(n(p))))
   | -P(i(i(p,q),i(n(q),n(p)))) | -P(i(i(p,i(q,r)),i(q,i(p,r))))
   | $ANS(step_allFrege_18_35_39_40_46_21).  % 21 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | -P(i(i(q,r),i(i(p,q),i(p,r))))
   | -P(i(p,i(n(p),q))) | -P(i(i(p,q),i(i(n(p),q),q))) | -P(i(i(p,i(p,q)),i(p,q)))
   | $ANS(step_allHilbert_18_21_22_3_54_30).  % 30 is dependent.
-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_allChurch_18_35_49).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r))))
   | $ANS(step_allLuka_19_37_59).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r)))))
   | $ANS(step_allWos_19_37_60).
-P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q)))
   | $ANS(step_allLuka_1_2_3).
end_of_list.


list(sos).
% Following is Meredith's axiom.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
end_of_list.

list(passive).
% Following are the Lukasiewicz three axioms.
-P(i(i(p,q),i(i(q,r),i(p,r)))) | $ANS(step_L1).
-P(i(i(n(p),p),p)) | $ANS(step_L2).
-P(i(p,i(n(p),q))) | $ANS(step_L3).
% Following are Desired associativity lemmas
-P(i(i(i(a,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 are Four distributivity theorems for which proofs are already known
-P(i(n(i(a,n(i(i(b,c),c)))),i(i(n(i(a,n(b))),n(i(a,n(c)))),n(i(a,n(c)))))) | $ANS(3_43a).
-P(i(i(i(n(i(a,n(b))),n(i(a,n(c)))),n(i(a,n(c)))),n(i(a,n(i(i(b,c),c)))))) | $ANS(3_43b).
-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_44a).
-P(i(n(i(i(n(i(n(a),b)),n(i(n(a),c))),n(i(n(a),c)))),i(n(a),n(i(i(n(b),n(c)),n(c)))))) | $ANS(3_44b).
% Two unproven (but provable!) distributivity theorems
-P(i(n(i(i(n(a),n(i(i(b,c),c))),n(i(i(b,c),c)))),i(i(n(i(i(n(a),n(b)),n(b))),n(i(i(n(a),n(c)),
   n(c)))),n(i(i(n(a),n(c)),n(c)))))) | $ANS(K_A_dist_1).
-P(i(i(i(n(i(i(n(a),n(b)),n(b))),n(i(i(n(a),n(c)),n(c)))), n(i(i(n(a),n(c)),n(c)))),
   n(i(i(n(a),n(i(i(b,c),c))), n(i(i(b,c),c)))))) | $ANS(K_A_dist_2).
% three old favorites!
-P(i(i(i(a,b),i(a,c)),i(i(b,a),i(b,c)))) | $ANS(3_51).
-P(i(i(i(i(c,a),i(b,a)),i(b,c)),i(i(i(c,b),i(a,b)),i(a,c)))) | $ANS(star).
-P(i(i(i(a,b),i(b,a)),i(b,a))) | $ANS(MV_5).
% Following are negations of theses 4 through 71.
-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).
end_of_list.


list(demodulators).
%  (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).
end_of_list.
```

```
list(hot).
-P(i(x,y)) | -P(x) | P(y).
% Following is Meredith's axiom.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
end_of_list.
```

Meredith's Proof

```
-----> EMPTY CLAUSE at   0.17 sec ----> 54 [hyper,2,53,43,34] $ANSWER(Luka,[1,2,3]).

Length of proof is 41.  Level of proof is 30.

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

1 [] -P(i(x,y)) |  -P(x) | P(y).
2 [] -P(i(i(p,q),i(i(q,r),i(p,r)))) |  -P(i(i(n(p),p),p)) |  -P(i(p,i(n(p),q))) |
$ANSWER(Luka,[1,2,3]).
3 [] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
4 [hyper,1,3,3] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
5 [hyper,1,3,4] P(i(i(i(x,i(n(y),z)),u),i(y,u))).
6 [hyper,1,3,5] P(i(i(i(x,x),y),i(z,y))).
7 [hyper,1,6,6] P(i(x,i(y,i(z,z)))).
9 [hyper,1,3,7] P(i(i(i(x,i(y,y)),z),i(u,z))).
11 [hyper,1,3,9] P(i(i(i(x,y),z),i(y,z))).
13 [hyper,1,11,3] P(i(x,i(i(x,y),i(z,y)))).
14 [hyper,1,11,13] P(i(x,i(i(i(y,x),z),i(u,z)))).
15 [hyper,1,13,5] P(i(i(i(i(i(x,i(n(y),z)),u),i(y,u)),v),i(w,v))).
16 [hyper,1,3,14] P(i(i(i(i(i(x,i(i(i(y,z),i(n(u),n(v))),u)),w),i(v6,w)),y),i(v,y))).
17 [hyper,1,3,15] P(i(i(i(x,y),i(z,i(n(n(y)),u))),i(v,i(z,i(n(n(y)),u))))).
18 [hyper,1,3,16] P(i(i(i(x,y),i(z,i(i(i(y,u),i(n(v),n(x))),v))),i(w,i(z,i(i(i(y,u),i(n(v),n(x))),v))))).
19 [hyper,1,17,3] P(i(x,i(i(y,z),i(n(n(y)),z)))).
20 [hyper,1,18,13] P(i(x,i(i(i(y,z),u),i(i(i(z,v),i(n(u),n(y))),u)))).
21 [hyper,1,19,19] P(i(i(x,y),i(n(n(x)),y))).
22 [hyper,1,3,19] P(i(i(i(i(x,y),i(n(n(x)),y)),z),i(u,z))).
23 [hyper,1,20,20] P(i(i(i(x,y),z),i(i(i(y,u),i(n(z),n(x))),z))).
25 [hyper,1,6,21] P(i(x,i(n(n(y)),y))).
26 [hyper,1,23,14] P(i(i(i(x,y),i(n(i(i(i(z,i(u,x)),v),i(w,v))),n(u))),i(i(i(z,i(u,x)),v),i(w,v)))).
27 [hyper,1,13,25] P(i(i(i(x,i(n(n(y)),y)),z),i(u,z))).
28 [hyper,1,3,26] P(i(i(i(i(i(x,i(y,i(z,u))),v),i(w,v)),z),i(v6,z))).
29 [hyper,1,3,28] P(i(i(i(x,y),i(z,i(u,i(y,v)))),i(w,i(z,i(u,i(y,v)))))).
30 [hyper,1,29,3] P(i(x,i(i(y,i(y,z)),i(u,i(y,z))))).
31 [hyper,1,30,30] P(i(i(x,i(x,y)),i(z,i(x,y)))).
32 [hyper,1,31,31] P(i(x,i(i(y,i(y,z)),i(y,z)))).
33 [hyper,1,32,32] P(i(i(x,i(x,y)),i(x,y))).
34 [hyper,1,5,33] P(i(x,i(n(x),y))).
35 [hyper,1,33,27] P(i(i(i(x,i(n(n(y)),y)),z),z)).
36 [hyper,1,33,22] P(i(i(i(i(x,y),i(n(n(x)),y)),z),z)).
38 [hyper,1,21,35] P(i(n(n(i(i(x,i(n(n(y)),y)),z))),z)).
39 [hyper,1,3,36] P(i(i(n(x),x),i(y,x))).
42 [hyper,1,14,38] P(i(i(i(x,i(n(n(i(i(y,i(n(n(z)),z)),u))),u)),v),i(w,v))).
43 [hyper,1,33,39] P(i(i(n(x),x),x)).
44 [hyper,1,3,42] P(i(i(i(x,n(i(i(y,i(n(n(z)),z)),n(u)))),v),i(u,v))).
45 [hyper,1,44,43] P(i(x,n(i(i(y,i(n(n(z)),z)),n(x))))).
46 [hyper,1,14,45] P(i(i(i(x,i(y,n(i(i(z,i(n(n(u)),u)),n(y))))),v),i(w,v))).
48 [hyper,1,33,46] P(i(i(i(x,i(y,n(i(i(z,i(n(n(u)),u)),n(y))))),v),v)).
49 [hyper,1,3,48] P(i(i(x,y),i(i(i(z,i(n(n(u)),u)),n(n(x))),y))).
50 [hyper,1,49,49] P(i(i(i(x,i(n(n(y)),y)),n(n(i(z,u)))),i(i(i(v,i(n(n(w)),w)),n(n(z))),u))).
51 [hyper,1,3,50] P(i(i(i(i(i(x,i(n(n(y)),y)),n(n(z))),u),v),i(i(z,u),v))).
```

```
53 [hyper,1,51,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
54 [hyper,2,53,43,34] $ANSWER(Luka,[1,2,3]).

------------ end of proof -------------
```

A 76-Step Proof, Free of Double Negation, for the Meredith Single Axiom

```
-----> EMPTY CLAUSE at  10.09 sec ----> 354 [hyper,7,331,287,132]
$ANS(step_allLuka_1_2_3).

Length of proof is 76.  Level of proof is 48.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
7 [] -P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) |
$ANS(step_allLuka_1_2_3).
8 [] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
53 [] -P(i(x,y)) | -P(x) | P(y).
54 [] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
55 [hyper,1,8,8] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
56 (heat=1) [hyper,53,54,55] P(i(i(i(x,i(n(y),z)),u),i(y,u))).
57 (heat=2) [hyper,53,54,56] P(i(i(i(x,x),y),i(z,y))).
60 [hyper,1,57,57] P(i(x,i(y,i(z,z)))).
61 (heat=1) [hyper,53,54,60] P(i(i(i(x,i(y,y)),z),i(u,z))).
62 (heat=2) [hyper,53,54,61] P(i(i(i(x,y),z),i(y,z))).
64 (heat=3) [hyper,53,62,54] P(i(x,i(i(x,y),i(z,y)))).
66 [hyper,1,62,64] P(i(x,i(i(i(y,x),z),i(u,z)))).
68 [hyper,1,56,64] P(i(x,i(i(i(y,i(n(x),z)),u),i(v,u)))).
71 (heat=1) [hyper,53,54,66] P(i(i(i(i(i(x,i(i(i(y,z),i(n(u),n(v))),u)),w),i(v6,w)),y),i(v,y))).
75 (heat=2) [hyper,53,54,71] P(i(i(i(x,y),i(z,i(i(i(y,u),i(n(v),n(x))),v))),i(w,i(z,i(i(i(y,u),i(n(v),n(x))),v))))).
84 [hyper,1,75,64] P(i(x,i(i(i(y,z),u),i(i(i(z,v),i(n(u),n(y))),u)))).
85 (heat=1) [hyper,53,84,54] P(i(i(i(x,y),z),i(i(i(y,u),i(n(z),n(x))),z))).
86 (heat=2) [hyper,53,85,54] P(i(i(i(x,y),i(n(i(i(x,z),i(u,z))),n(i(i(i(z,v),i(n(w),n(u))),w)))),i(i(x,z),i(u,z)))).
97 [hyper,1,85,66] P(i(i(i(x,y),i(n(i(i(i(z,i(u,x)),v),i(w,v))),n(u))),i(i(i(z,i(u,x)),v),i(w,v)))).
98 [hyper,1,85,64] P(i(i(i(x,y),i(n(i(i(i(z,x),u),i(v,u))),n(z))),i(i(i(z,x),u),i(v,u)))).
99 (heat=1) [hyper,53,54,97] P(i(i(i(i(i(x,i(y,i(z,u))),v),i(w,v)),z),i(v6,z))).
100 (heat=1) [hyper,53,54,98] P(i(i(i(i(i(x,i(y,z)),u),i(v,u)),y),i(w,y))).
101 (heat=2) [hyper,53,54,99] P(i(i(i(x,y),i(z,i(u,i(y,v)))),i(w,i(z,i(u,i(y,v)))))).
102 (heat=2) [hyper,53,54,100] P(i(i(i(x,y),i(z,i(y,u))),i(v,i(z,i(y,u))))).
103 (heat=3) [hyper,53,101,54] P(i(x,i(i(y,i(y,z)),i(u,i(y,z))))).
105 (heat=4) [hyper,53,103,54] P(i(i(x,i(x,y)),i(z,i(x,y)))).
106 [hyper,1,62,98] P(i(i(n(i(i(i(x,y),z),i(u,z))),n(x)),i(i(i(x,y),z),i(u,z)))).
107 [hyper,1,102,68] P(i(x,i(i(i(y,i(n(i(z,u)),v)),w),i(u,w)))).
108 [hyper,1,102,66] P(i(x,i(i(i(y,i(z,u)),v),i(u,v)))).
109 (heat=1) [hyper,53,54,107] P(i(i(i(i(i(x,i(n(i(y,z)),u)),v),i(z,v)),w),i(v6,w))).
111 (heat=1) [hyper,53,54,108] P(i(i(i(i(i(x,i(y,z)),u),i(z,u)),v),i(w,v))).
113 (heat=2) [hyper,53,54,109] P(i(i(i(x,y),i(z,i(n(i(u,n(y))),v))),i(w,i(z,i(n(i(u,n(y))),v))))).
117 (heat=3) [hyper,53,113,54] P(i(x,i(i(y,z),i(n(i(u,n(y))),z)))).
119 (heat=4) [hyper,53,117,54] P(i(i(x,y),i(n(i(z,n(x))),y))).
121 [hyper,1,105,105] P(i(x,i(i(y,i(y,z)),i(y,z)))).
124 (heat=1) [hyper,53,121,54] P(i(i(x,i(x,y)),i(x,y))).
125 [hyper,1,102,106] P(i(x,i(i(i(y,z),u),i(n(y),u)))).
126 (heat=1) [hyper,53,125,54] P(i(i(i(x,y),z),i(n(x),z))).
132 [hyper,1,56,124] P(i(x,i(n(x),y))).
135 [hyper,1,124,61] P(i(i(i(x,i(y,y)),z),z)).
148 [hyper,1,56,132] P(i(x,i(n(i(y,i(n(x),z))),u))).
162 [hyper,1,86,148] P(i(i(x,y),i(n(i(x,z)),y))).
```

```
174 [hyper,1,119,162] P(i(n(i(x,n(i(y,z)))),i(n(i(y,u)),z))).
186 [hyper,1,105,174] P(i(x,i(n(i(y,n(i(y,z)))),z))).
204 [hyper,1,64,186] P(i(i(i(x,i(n(i(y,n(i(y,z)))),z)),u),i(v,u))).
224 [hyper,1,124,204] P(i(i(i(x,i(n(i(y,n(i(y,z)))),z)),u),u)).
225 (heat=1) [hyper,53,54,224] P(i(i(i(x,n(i(x,n(y)))),z),i(y,z))).
227 [hyper,1,111,225] P(i(x,i(y,i(z,n(i(i(u,i(v,z)),n(y))))))).
228 [hyper,1,57,225] P(i(x,i(y,i(z,n(i(z,n(y))))))).
229 [hyper,1,225,126] P(i(x,i(n(y),n(i(i(y,z),n(x)))))).
231 (heat=1) [hyper,53,54,227] P(i(i(i(x,i(y,n(i(i(z,i(u,y)),n(x))))),v),i(w,v))).
232 (heat=1) [hyper,53,228,54] P(i(x,i(y,n(i(y,n(x)))))).
233 (heat=2) [hyper,53,54,231] P(i(i(i(x,y),z),i(i(i(u,i(v,n(y))),n(i(z,w))),z))).
234 (heat=3) [hyper,53,233,54] P(i(i(i(x,i(y,n(z))),n(i(i(i(z,u),i(v,u)),w))),i(i(z,u),i(v,u)))).
235 (heat=4) [hyper,53,54,234] P(i(i(i(i(x,y),i(z,y)),u),i(x,u))).
241 [hyper,1,57,232] P(i(x,i(y,n(i(y,n(i(z,z))))))).
243 (heat=1) [hyper,53,241,54] P(i(x,n(i(x,n(i(y,y)))))).
245 [hyper,1,233,64] P(i(i(i(x,i(y,n(z))),n(i(i(i(i(u,z),v),i(w,v)),v6))),i(i(i(u,z),v),i(w,v)))).
246 (heat=1) [hyper,53,54,245] P(i(i(i(i(i(x,y),z),i(u,z)),v),i(y,v))).
247 (heat=2) [hyper,53,246,54] P(i(i(n(x),n(y)),i(i(i(z,x),u),i(y,u)))).
250 [hyper,1,235,124] P(i(x,i(i(x,y),y))).
265 [hyper,1,250,241] P(i(i(i(x,i(y,n(i(y,n(i(z,z)))))),u),u)).
267 [hyper,1,250,229] P(i(i(i(x,i(n(y),n(i(i(y,z),n(x))))),u),u)).
268 (heat=1) [hyper,53,54,265] P(i(i(x,y),i(i(n(x),n(i(z,z))),y))).
269 (heat=1) [hyper,53,54,267] P(i(i(x,y),i(i(i(x,z),n(i(y,u))),y))).
279 [hyper,1,268,250] P(i(i(n(x),n(i(y,y))),i(i(x,z),z))).
281 [hyper,1,269,279] P(i(i(i(i(n(x),n(i(y,y))),z),n(i(i(i(x,u),u),v))),i(i(x,u),u))).
283 (heat=1) [hyper,53,54,281] P(i(i(i(i(x,y),y),n(x)),i(z,n(x)))).
284 [hyper,1,124,283] P(i(i(i(i(x,y),y),n(x)),n(x))).
285 (heat=1) [hyper,53,54,284] P(i(i(n(x),x),i(y,x))).
286 [hyper,1,126,284] P(i(n(i(i(x,y),y)),n(x))).
287 [hyper,1,124,285] P(i(i(n(x),x),x)).
290 [hyper,1,247,286] P(i(i(i(x,i(i(y,z),z)),u),i(y,u))).
295 [hyper,1,290,243] P(i(x,n(i(i(y,i(i(x,z),z)),n(i(u,u)))))).
302 [hyper,1,234,295] P(i(i(x,y),i(i(i(z,i(u,n(x))),y),y))).
315 [hyper,1,302,135] P(i(i(i(x,i(y,n(i(i(z,i(u,u)),v)))),v),v)).
318 (heat=1) [hyper,53,54,315] P(i(i(x,y),i(i(i(z,i(u,u)),x),y))).
326 [hyper,1,318,318] P(i(i(i(x,i(y,y)),i(z,u)),i(i(i(v,i(w,w)),z),u))).
329 (heat=1) [hyper,53,54,326] P(i(i(i(i(i(x,i(y,y)),z),u),v),i(i(z,u),v))).
331 (heat=2) [hyper,53,329,54] P(i(i(x,y),i(i(y,z),i(x,z)))).
354 [hyper,7,331,287,132] $ANS(step_allLuka_1_2_3).

------------ end of proof -------------
```

References

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

[Rose1958] Rose, A., and Rosser, J. B., "Fragments of many-valued statement calculi", *Trans. AMS* **87** (1958) 1–53.

[Veroff1996] Veroff, R., "Using hints to increase the effectiveness of an automated reasoning program: Case studies", *J. Automated Reasoning***16**, no. 3 (1996) 223–239.

[Wos1996] Wos, L., *The Automation of Reasoning: An Experimenter's Notebook with OTTER Tutorial,* Academic Press, New York, 1996.

[Wos1998] Wos, L., "Automating the search for elegant proofs", *J. Automated Reasoning* **21,** no. 2 (1998) 135–175.

[Wos1999a] Wos, L., with Pieper, G., *A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning,* World Scientific, 1999.

[Wos1999b] Wos, L., with Pieper, G., "The hot list strategy", *J. Automated Reasoning* **22,** no. 1 (1999) 1–44.

[Wos1999c] Wos, L., "Lemma inclusion versus lemma adjunction", *AAR Newsletter* **44**, July 1999 (http://www.mcs.anl.gov/AAR/issuejuly99/index.html).