Branden Fitelson
Larry Wos

Finding Missing Proofs with Automated Reasoning*

Abstract. This article features long-sought proofs with intriguing properties (such as the absence of double negation and the avoidance of lemmas that appeared to be indispensable), and it features the automated methods for finding them. The theorems of concern are taken from various areas of logic that include two-valued sentential (or propositional) calculus and infinite-valued sentential calculus. Many of the proofs (in effect) answer questions that had remained open for decades, questions focusing on axiomatic proofs. The approaches we take are of added interest in that all rely heavily on the use of a single program that offers logical reasoning, William McCune's automated reasoning program OTTER. The nature of the successes and approaches suggests that this program offers researchers a valuable automated assistant. This article has three main components. First, in view of the interdisciplinary nature of the audience, we discuss the means for using the program in question (OTTER), which flags, parameters, and lists have which effects, and how the proofs it finds are easily read. Second, because of the variety of proofs that we have found and their significance, we discuss them in a manner that permits comparison with the literature. Among those proofs, we offer a proof shorter than that given by Meredith and Prior in their treatment of Łukasiewicz's shortest single axiom for the implicational fragment of two-valued sentential calculus, and we offer a proof for the Łukasiewicz 23-letter single axiom for the full calculus. Third, with the intent of producing a fruitful dialogue, we pose questions concerning the properties of proofs and, even more pressing, invite questions similar to those this article answers.

Keywords: missing proofs, axiomatic proofs, logic calculi, condensed detachment, term-avoidance proofs, automated reasoning.

1. Basic tenets, some background, and an illustration

Consistent with the emphasis of the great researchers that include Hilbert, Łukasiewicz, Meredith, Prior, and Wajsberg, this article focuses on axiomatic proofs. However, in contrast to their work that yielded one significant proof after another purely through the exercise of the mind, the proofs we feature are obtained with indispensable assistance from a computer program that applies logical reasoning, William McCune's program OTTER [McCune1994]. As evidence of the value of reliance on this program, we note that many of the proofs we have found were missing from the extensive literature of logic. An excellent example of such a missing proof concerns the Łukasiewicz 23-letter single axiom for two-valued (or propositional) calculus.

In addition to finding missing proofs — in the tradition of Meredith and Prior, Thomas, Ulrich, and others — OTTER has proved invaluable in finding shorter proofs. Our success in finding a proof (shorter than that of Meredith and Prior) for the Łukasiewicz shortest single axiom for the implicational fragment of two-valued sentential calculus illustrates the value of reliance on an automated reasoning program.

In this article, in addition to the focus on various proofs, we discuss both the manner in which OTTER was used and the arsenal of weapons it offers. In that regard, as a brief preview, we provide (prior to Section 2) a taste of how one can proceed.

1.1. Basic tenets

One of the basic tenets on which this article rests asserts that axiomatic proofs are preferable to other types, such as those based on metatheory or on the presence of the property of validity (a model-theoretic argument). In our view, access to an axiomatic proof (compared with some other type of proof) is more likely to lead both to the discovery of proofs that have been missing from the literature and to the formulation of some new useful approach. In general, we find axiomatic proofs to be more instructive.

That type of proof also offers an appeal not directly germane to logic or mathematics, namely, that such proofs are at least theoretically well within reach of a general-purpose automated reasoning program. As evidence of the truth of this observation, this article features not only proofs absent from the literature, but proofs that answer questions that had remained open for decades, questions focusing on axiomatics. Especially from the perspective of the logician or mathematician who is primarily interested in new results, these proofs (presented in Section 4) are the meat of this article.

For a preview of what is to come in this regard, we note that in 1936 Łukasiewicz offered the following 23-letter single axiom for two-valued sentential (or propositional) calculus [Łukasiewicz1970, page 225, footnote 10], where the function i denotes logical *implication* and the function n denotes logical *negation*.

```
i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))
```

What made this announcement particularly intriguing to researchers was the absence both of a proof and of any hint as to its nature. Further, from what

* 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.

Presented by **Jacek Malinowski**; *June* March 17, 2000

we know, the literature offers no aid to finding such a proof. Our goal was that of finding an axiomatic proof, one relying on the often-used inference rule *condensed detachment* (discussed in Section 2).

This article offers the desired proof (in Section 4.2), presents a new methodology that yielded the proof through heavy use of OTTER, and focuses on the presence of a property that is indeed intriguing and perhaps seldom studied, namely, the absence of any term of the form $n(n(t))$ for any term t. This double-negation-free property is the focus in Section 5 of a question whose answer is currently unknown.

This article also offers a proof that addresses a different aspect, namely, that of proof improvement. Specifically, Meredith and Prior [Meredith1963, page 171] present a "very slight abridgement" of Łukasiewicz's proof [Łukasiewicz1970, pages 299–300] of the sufficiency of his shortest single axiom for the implicational fragment of two-valued sentential (or propositional) calculus. We give in Section 4.2 a further abridgement.

In addition to the cited tenet, we take the position that the combination of researcher and reasoning program offers far more than the sum of its parts. Indeed, a means exists for the researcher to impart knowledge, experience, and intuition to the program, and the program in turn can apply techniques (featuring generality of deduction) that are at least most uncomfortable for the researcher. The program's emphasis on generality will be nicely illustrated (in Section 2) when this article focuses on a four-step proof that (some logicians have said) would never have been discovered by an unaided researcher. The interplay of the researcher's skills with the assets of a powerful reasoning program yields far more than either alone can yield.

One additional tenet completes the foundation for this presentation. Specifically, we hold that the answering of one open question is likely to lead to the answering of other open questions. In that regard, we pose questions whose answers have eluded us and, perhaps more important, invite the submission of questions similar to those featured here. For evidence of the asserted cascading of the answering of open questions, this article lists successes that are not all closely related nor all from the same area of logic but that were connected in time and in methodology.

1.2. Some background

For the background that addresses the interests of a diverse audience, that concerns the automation of logical reasoning, and that focuses on a useful program for research, (at the general level) we recommend two books [Wos1999,Wos2000b]. The topics covered in the first of the two books include a full treatment of the basic elements of automated reasoning, the formal foundations, various applications, and an offering of open questions. That book serves well as a text and also as a source for research topics. The second (two-volume) book is a compendium of research papers published between 1958 and 1998 inclusive. The majority of these papers focus on the developments of automated reasoning that include the introduction of new inference rules, new strategies for controlling the application of the inference rules, and the answering of various open questions.

Regarding specific background aspects, one can gain easy access to the program OTTER, which played such a key role in the successes featured in this article, by consulting the CD-ROM included in the first of the two cited books. On that CD-ROM, one also finds the information (in the form of a manual) for using OTTER, appropriate input files, and other pertinent items (some taken from relevant books). In this article, we devote Section 3 to the nature and use of various lists, flags, and parameters that affect the actions of OTTER. By doing so, we hope to provide insight into the reasons OTTER has proved so valuable and also enable the researcher to assess its value as an aid to finding proofs. As for the proofs OTTER finds, a neat proof (given in Section 2) will be used to illustrate how one reads them.

1.3. An illustration of interplay

For a preview of the interplay between researcher and reasoning program and the type of significant contribution to logic that can result, we focus (in Section 4.1) on an important theorem from infinite-valued sentential calculus. The theorem asserts that logical *or* is associative, where the *or* of x and y in terms of *implication* is $i(i(x, y), y)$. Rather than asking the program to attempt to prove the corresponding equality, the assignment consisted of attempting to prove the two appropriate implications. (We note that OTTER is well equipped to cope with the equality relation directly, far more capable than many reasoning programs. However, because of historical precedent concerning the use of condensed detachment, because of our formulation of powerful relevant methodologies, and because we were certain that the avoidance of the equality relation would, in this case and others like it, contribute markedly to effectiveness, we chose the cited approach.)

The most obvious role played by the researcher is that of choosing the theorem to attempt to prove and conveying that assignment to the program. Indeed, the program "knows" almost nothing and, if left on its own, is unable to recognize the importance of one conclusion when compared with another. To enable the program to detect and then signal that a proof

has been completed, the researcher includes in the problem description the assumption that the conclusion does not hold, that the theorem is false. The program has the means to compare conclusions pairwise and recognize, if such is the case, that a contradiction has been found.

A proof already existed in the literature establishing logical *or* to be associative in many-valued sentential calculus. However, as far as we knew when we studied that theorem, no purely axiomatic proof was available. Further, we were unable to provide any suggestions about the actual proof steps (that might be present in an axiomatic proof) when we commenced our attack with OTTER. Two formulas were to be deduced, whose respective negations are given with the following two clauses, where "-" denotes logical **not** and where the predicate P denotes "provable".

```
-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)))).
```

With the details left to later, we note that OTTER quickly proved one of the two desired implications (the second), but it appeared to make no progress with the other. (The program on its own is unable to measure its progress, its closeness to reaching the objective.) For the first implication, in contrast to our attack on the second, we chose to include suggestions regarding a possible proof. We (in effect) instructed the program to emphasize the role of any new conclusion it retained that was similar (where all variables are treated as indistinguishable) to one of the proof steps found in its proof of the second implication. That apparently naive instruction was just what was needed. Indeed, the program quickly found the desired proof.

One more aspect of the research in focus merits mention. As is so typical of our studies, we then sought proofs of the two implications shorter than those we had in hand. That effort not only succeeded but, and here is where the generality property of the program's reasoning came into play with a rather startling result, two implications were proved, each more general than its counterpart corresponding to the original goal. Whether the added generality of the theorem that was proved is of much interest is left to the audience to assess. Independent of the verdict, in place of the metatheoretic proof, we now have an axiomatic proof of the associativity of *or* in infinite-valued sentential calculus, which was our goal, and an axiomatic proof of a more general theorem. We shall give in Section 4.1 the proofs of the more general implications.

2. Reading automated proofs

A few items regarding notation, conventions, and inference rules set the stage for reading the proofs produced by a program such as OTTER.

The type of proof yielded by a program such as OTTER has one singular advantage over most proofs found in the literature. Specifically, this type of proof explicitly gives, for each deduced step, the precise history concerning which hypotheses were used and what inference rule was employed. Other than when canonicalization and simplification are in use, no steps are present implicitly, as is often the case in the literature. However, even with the precise derivation information in hand, more than occasionally one encounters a proof step that produces puzzlement. As will be illustrated, the cause for the occasional obscurity rests with the program's insistence on generality of reasoning. In addition to the cited possible obstacle to proof reading, as expected, a notational problem may be a bit annoying.

In that regard, the following small number of notational conventions merit mention. For example, if one wishes OTTER to study the earlier-cited Łukasiewicz 23-letter formula, the following *clause* is included in the input to the program.

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

The predicate P (in the given clause) can be interpreted as "is provable" or as "is true". The *clause language*, which is a dialect of first-order predicate calculus, is that which is used to present a question or problem to the program.

Regarding the remaining notation (needed for this article), logical *or* is denoted by "|" and logical *not* by "-". Logical *and* is present implicitly between each pair of clauses. No other logical connectives are permitted in the clause language. The scope of a variable is limited to the clause in which it occurs, and all variables are implicitly universally quantified. For OTTER, a variable is denoted by an expression beginning with a letter between lower-case u and z inclusive.

To further illustrate the use of the clause language, we focus on the inference rule *condensed detachment* employed in the proofs given in this article. Condensed detachment is frequently used in the literature of logic and is quite reminiscent of modus ponens. Condensed detachment considers two formulas, $i(a, b)$ (the major premiss) and c (the minor premiss), and, if c unifies with a, yields the formula d, where d is obtained by applying to b the most general unifier of c and a. In other words, to apply the rule successfully, a and c must unify — a substitution must exist that, when

applied, causes a and c to become identical. The program always seeks the most general unification. For the program to apply condensed detachment, one includes the following clause and chooses as the inference rule that called *hyperresolution*.

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

For example, if condensed detachment is applied to the first two of the following three clauses, the third is yielded, where the second clause is playing the role of c (the minor premiss).

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

If the roles of the two formulas (expressed as clauses) are reversed and condensed detachment is applied, a copy of the first formula (expressed as a clause) is obtained.

We now turn to a proof of some substance; indeed, it establishes the deducibility of one three-axiom system of two-valued sentential calculus from another three-axiom system. In the following proof, the first clause is that for condensed detachment, and the next three are the respective clause equivalents of one of the two axiom systems, that due to Wos. The fifth clause is the negation of a member of the second axiom system; the two systems share in common two elements, theses 19 and 37. Therefore, all that is needed is a deduction of thesis 59 from the members of the first system, theses 19, 37, and 60. As for the four deduced steps, each is numbered according to its place among the deduced conclusions. The triple that appears after "hyper" follows the order first, the condensed detachment clause, second, the clause that plays the role of major premiss, and third, the clause that plays the role of the minor premiss. After giving the proof, we shall discuss in detail its least obvious step, the last. Such steps can indeed present a problem because of relying on a type of reasoning more suited to a computer than to a researcher.

A Neat Proof Focusing on the Wos Axiom System for Two-Valued Sentential Calculus

```
5 [] -P(i(x,y)) | -P(x) | P(y).
6 [] -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(negation_thesis_59).
7 [] P(i(i(i(x,y),z),i(y,z)))   # label(thesis_19).
8 [] P(i(i(i(x,y),z),i(n(x),z)))   # label(thesis_37).
```



```
9 [] P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z)))))   # label(thesis_60).
------------------------------------
16 [hyper,5,9,8] P(i(i(i(x,y),z),i(i(u,z),i(i(x,u),z)))).
23 [hyper,5,16,7] P(i(i(x,i(y,z)),i(i(i(u,y),x),i(y,z)))).
30 [hyper,5,23,7] P(i(i(i(x,y),i(i(z,y),u)),i(y,u))).
34 [hyper,5,30,9] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))) # label(thesis_59).
```

Clause (34) contradicts clause (6), and the proof is complete.

Fortunately, if a proof step is obscure or its soundness is doubted, OTTER offers what is needed. Indeed, the use of the option set(build_proof_object) produces a file that presents all of the substitutions for all of the variables on which each condensed detachment step rests.

Let us examine the last step of the four-step proof, detailing the substitutions of terms for variables that one learns of with the use of build_proof_object. We choose that step because it is the least obvious. More generally, with condensed detachment, when a deduction relies upon a nontrivial substitution of terms for variables in *both* premisses, the corresponding step can even cause one to doubt its soundness. Technically, that type of nontrivial substitution is called a *two-way unification*, in contrast to a one-way unification. The first three steps of the four-step proof require only a one-way unification, merely a substitution of terms for variables in but one of the two premisses. Two examples nicely set the stage for the discussion of the fourth step of the four-step proof.

If one is told that Plato likes everybody and also told that Plato does not like Ari, then one has been presented with a pair of contradictory statements. The following two clauses capture the example.

```
LIKES(Plato,x)
-LIKES(Plato,Ari).
```

A program such as OTTER immediately detects the contradiction by noting that a one-way unification exists, the substitution of Ari for x in the first clause, without actually producing the corresponding instantiated clause for retention.

A bit more interesting is the second case in which one is told that Plato likes everybody and also told that nobody likes Plato. Again, two clauses suffice.

```
LIKES(Plato,x)
-LIKES(y,Plato).
```

For the program to detect the contradiction (in the second case), a two-way unification is required, a substitution of Plato for x in the first clause together

with a substitution of Plato for y in the second. In contrast to the simplicity of the given two-way unification, as the following shows, the substitution required to deduce the fourth step of the four-step proof is rather complex. Note that the substitutions found by the program when trying to unify two expressions are the most general that it can find that identify a common domain, that make the two expressions being unified identical.

The fourth step, numbered 34, is obtained by applying condensed detachment to the step numbered 30 as major premiss and 9 as minor premiss. To make the unification more transparent, rename the variables in 30, respectively, to x1, y1, z1, and u1, with the variables in 9 remaining u, x, z, and y. The desired substitution of terms for variables in both clauses is the following (applied simultaneously to the listed variables): in 30, i(x,i(n(y),z)) for x1, i(n(y),z) for y1, x for z1, and i(i(u,z),i(i(y,u),z)) for u1; and in 9, i(x,i(n(y),z)) for u, y for x, z for z, and u for y.

Such complex substitutions often occur in a proof completed by a reasoning program, which explains in part why that type of program finds proofs that have been missing for decades and (perhaps) why unassisted researchers did not find them. Some of the new proofs (as we discuss in Section 4) have no counterpart in the literature, as in the case of the Łukasiewicz 23-letter single axiom. Of a different cast, some of the proofs we have found do have a correspondent in the literature, but the new proof we offer is more elegant in one or more senses that include proof length and term structure. The finding of these new proofs (in part because of the emphasis on generality of reasoning by the program) nicely demonstrates the value of combining the expertise of a researcher with the approach taken by a program such as OTTER. To further illustrate how such combinations take place, we turn to various aspects of OTTER.

3. Lists, flags, and parameters

We conjecture that the results pertinent to areas of logic and that are featured later in this article would not have been obtained without access to OTTER's impressive arsenal of weapons. Also, we conjecture openly that the researcher adding the use of this program in the attack on an open question will greatly increase the likelihood of finding the answer. Because of these two factors, in this section we focus on what OTTER offers, rather than discussing some generic automated reasoning program. Of prime concern here is the interplay between researcher and program.

An effective interplay of researcher and reasoning program begins with a judicious use of lists. The placement in the various lists of the clauses that

present the question or problem to the program influences the program's performance sharply and, equally important, often more than compensates for the program's lack of knowledge, experience, and intuition. Indeed, just as a researcher restricts the reasoning applied in the attempt to reach an objective, the program can do the same. The *set of support list* is the list in which the researcher is well advised to place the clauses that are conjectured to be used for inference rule initiation at the beginning of the program's attack.

For example, reminiscent of what a person might do when seeking a proof of the theorem that asserts that rings in which the cube of x is x are commutative, a good move is to place the clause equivalent of $xxx = x$ in the set of support list and place the axioms in a list called *usable*. This move nicely prepares the way for the program to make effective use of the powerful restriction strategy known as the *set of support strategy*. The strategy prevents the program from applying an inference rule to a set of hypotheses all of which are input clauses in the usable list, drawing conclusions only if recursively traceable to the input set of support. The program is thus prevented from exploring the entire underlying theory from which the specific theorem of concern is taken, which in turn markedly reduces the size of the search space of conclusions to consider. In contrast, a researcher often does apply an inference rule to sets of axioms without the worry of getting lost or producing too much new information.

As for the assumption that the conclusion is false (that such a ring is not commutative), which prepares the way for seeking a proof by contradiction, its clause equivalent can be placed in the set of support list or in the usable list or, if one wishes this clause to participate only for determining that the proof has been found, in the *passive list*. Members of the passive list are used either to detect proof completion or to purge a type of redundant information. For a second example of the use of the set of support strategy, a good move for studying the type of theorem featured in this article has one place the axioms (say, for infinite-valued sentential calculus) in the set of support list, the clause for condensed detachment in the usable list, and the denial of the theorem either in the passive list or in the usable list.

Although essentially no restriction of the reasoning results in this case, other actions with another list can be taken to restrict it. In particular, one of the *weight lists* can be used to prevent the program from retaining information deemed unnecessarily complex, the means being to assign to the appropriate parameter a value that the program treats as an upper bound on complexity.

Further interplay between researcher and program is enabled by use of the *demodulator list*. This list contains the equalities that the program uses

for simplification and canonicalization. Such equalities can be chosen by the researcher, adjoined by the program during a run, or both.

In addition to restricting the program's reasoning, directing it is often crucial. Again, a weight list can be of assistance. Indeed, as a fine illustration of much interplay between researcher and program, one can use such a list in conjunction with the *resonance strategy* to instruct the program to key on formulas or equations that are similar to, but frequently not identical to, ones conjectured by the researcher to merit emphasis. Such formulas or equations are called *resonators*, and to each the researcher can assign a value to reflect its conjectured relative significance for directing the program's reasoning. The resonators that are chosen enable the program to benefit from the researcher's knowledge, experience, and even intuition. The choice of resonators often is the key to finding a missing proof.

For a fine example, the resonance strategy played a vital role in the finding by OTTER of the first known (to us) proof establishing the Łukasiewicz 23-letter formula to be a single axiom for two-valued sentential calculus. Some small expansion of this fact is in order to illustrate how the combination of researcher and program can be effective, even when the relevant researchers were far from expert.

From what we know, the literature offers nothing that sheds any light on the nature of the proof Łukasiewicz had in mind. Quite likely (although not necessarily), the proof known to him relied mainly or exclusively on condensed detachment. This inference rule is known to be sufficient [Kalman1983]. As for the target of OTTER's attack, we included various axiom systems for this area of logic, conjecturing that the Łukasiewicz three-axiom system was the most easily reached and (perhaps) the one he used to complete his proof. Not because of any deep insight, but rather because of earlier successes, we included sixty-eight resonators, each corresponding to a theorem proved by Łukasiewicz, theorems he denotes as theses 4 through 71.

In one afternoon, a few hours of computer time spread over four runs sufficed to produce a 200-step proof, completing with the deduction of the cited three-axiom system. Of the deduced 200 steps, 8 are among the sixty-eight cited theses, and 22 match (where all variables are treated as indistinguishable) one of the included resonators. Although one can hardly claim that the number — just 22 — of steps matching a resonator is large among the 200 steps of the proof under discussion, we nevertheless conjecture that the program's keying on them was indispensable to the success.

Still in the context of the successful completion of the 200-step proof, one of OTTER's options demands mention because, without reliance on it, (we assert) the cited proof would have quite likely remained out of reach.

The option in question focuses on the type of term (found in deduced-and-retained conclusions) to be encouraged or discouraged. Specifically — and most intriguing to us — the deduced steps of the cited 200-step proof are free of double-negation terms, terms of the form $n(n(t))$ for some term t where the function n denotes negation. This aspect was by design: Indeed, we instructed OTTER to discard any deduced conclusion containing a double-negation term. This decision was based on prior successes in obtaining proofs of deep theorems in various areas of logic, proofs in which we required that double-negation terms be absent.

Again, the contrast (as well as the interplay) between researcher and program comes into play. In particular, blocking the use of such terms is apparently counterintuitive, if one bases an opinion on the literature. On the other hand, avoidance of conclusions containing such terms appears to sharply increase the density of useful information within that which is retained. In the context of interplay, other types of term can be easily avoided, if that is the wish of the researcher, whether prompted by efficiency considerations or prompted by the goal of completing a proof satisfying the corresponding constraint.

The aid that OTTER can provide is by no means limited to the preceding items. Among the other options the program offers (each governed by some flag or parameter) are the following. Indeed, various flags (if set or cleared) determine which inference rules are to be used, among which is one that treats equality as if it is "understood". One flag affects the type of search to be used, breadth first (first come first served) if set, and otherwise based on conclusion complexity. If the latter, the complexity can be measured purely in terms of symbol count, or it can be determined by user-supplied templates. One of OTTER's parameters can be used to assign an upper bound to the complexity of newly retained conclusions; deduced conclusions whose complexity exceeds the assigned value are purged. If the choice is to blend (to some extend) a breadth-first search with one based on conclusion complexity, the needed parameter is offered.

OTTER can be of even greater service. In particular, the reading of a proof in the literature more than occasionally leaves one with questions regarding the precise details concerning one or more proof steps. If the researcher so desires, for each completed proof, the setting of the appropriate flag instructs the program to provide all of the corresponding details, a proof_object. If, for example, OTTER produces a proof totally free of double negation, one might wish to know whether any of the intermediate substitutions (of terms for variables) relied upon in any of the condensed-detachment steps required the use of double negation. Yet another flag that

serves well in the context of condensed detachment is that which instructs the program to list the history of each deduced step in a manner that enables one to know which hypothesis was used as major premiss and which as minor.

The program OTTER can also be of assistance in the context of the quality of the proof it finds. Indeed — quite relevant to much of our research — some flags and parameters are directly pertinent to having OTTER search for proofs that are elegant in one or more aspects. In particular, regarding different paths to the same deduced conclusion, one flag has the program give preference to shorter deduction paths over longer. This option provides a fair amount of assistance when the goal is to find a proof shorter than that in hand. If the aspect of elegance of interest concerns the maximum number of distinct variables permitted in deduced proof steps, the program offers the appropriate parameter. The aspects of elegance focusing on the complexity of deduced steps and on term structure have already been discussed.

4. Beautiful old theorems and intriguing new proofs

By an "old theorem", we mean a result that has been known to hold for at least a decade. With that definition, the theorems featured here are very old, but very significant. Of a complementary nature, some proofs are so young that they have not yet been born or have only recently been born. In this article, we focus on recently born proofs, young proofs of very old and significant theorems.

Hilbert (whom some label as Mr. Axiom) might find our proofs (which we have found through the use of automated reasoning) most satisfying, for they are axiomatic in the strictest sense, with no steps left to the imagination. Also satisfying is that the proofs rely solely on the inference rule condensed detachment, with no recourse to instantiation or equality-oriented reasoning. In the given senses, the proofs are indeed pure, nicely in the strict spirit of the logic calculi of concern.

From another perspective, the proofs we have found with OTTER fill in gaps in the literature. For but one example, highly unsatisfying is the case in which a result is known to hold because of the proven fact that it is valid, but no axiomatic proof is offered in print. We feature such a case, as well as other types that lacked an axiomatic proof. We include a few proofs and list a number of our successes. For a more complete set of proofs, we recommend the forthcoming book *Automated Reasoning and the Finding of Missing and Elegant Proofs in Formal Logic*.

4.1. Infinite-valued sentential calculus

The area of logic of concern in this section is infinite-valued sentential calculus, sometimes called many-valued sentential calculus. Łukasiewicz provided the following axiom system (consisting of five formulas) for this field, expressed in clause notation.

```
%  The following four formulas are, respectively,
%  MV1 through MV4.
P(i(x,i(y,x))).
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(i(x,y),y),i(i(y,x),x))).
P(i(i(n(x),n(y)),i(y,x))).
%  The following formula is MV5.
P(i(i(i(x,y),i(y,x)),i(y,x))).
```

Some years later Meredith [Meredith1958] proved that the fifth of the five formulas is dependent on the first four (a far-from-obvious fact). In this article, we give a more elegant proof found by OTTER, a proof that is shorter than Meredith's and that has the intriguing property that no terms of the form $n(n(t))$ are present for any term t, where the function n denotes negation.

The first theorem whose proof we give asserts that logical *or* is associative, where the *or* of x and y in terms of *implication* is $i(i(x, y), y)$. However, rather than proving the theorem as stated, we prove a generalization of it, which then leads to one of the promised questions we have for logicians. Specifically, rather than proving

P(i(i(i(x,i(i(y,z),z)),i(i(y,z),z)),i(i(i(i(x,y),y),z),z))),

we prove the more general formula

P(i(i(i(x,i(i(y,z),z)),i(i(y,z),u)),i(i(i(i(x,y),y),z),u))),

and, rather than proving

P(i(i(i(i(i(x,y),y),z),z),i(i(x,i(i(y,z),z)),i(i(y,z),z)))),

we prove the more general formula

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

- QUESTION. Are the more general implications of logical significance compared with the two that capture associativity?

Before we give the two proofs respectively of the more general formulas, a few remarks are in order. First, we know of no shorter proofs than the two we give. Second, both proofs avoid double negation. In fact, the second

proof avoids negation entirely, but the first requires the use of the fourth axiom of Łukasiewicz given earlier. Third, one can find a proof of the so-called first associative formula in which the variable u is absent by relying solely on the first three Łukasiewicz axioms. Fourth, in the context of the preceding bulleted question, we have no estimate of the significance of the two generalizations.

The triple that appears after the word "hyper" follows the order first, the condensed detachment clause, second, the clause that plays the role of major premiss, and third, the clause that plays the role of the minor premiss. The first number, j, that appears in a proof step gives the corresponding position of the clause within those retained by OTTER. In particular, the number of the last clause listed denotes the number of clauses that were retained when the proof was completed. For example, the designation 13466 of the last step in the following proof denotes that 13466 clauses were retained when the proof was completed. Next, when OTTER is instructed to use a strategy known as the *hot list strategy*, one may find an input clause appearing more than once in the input of a proof. In that regard, the designation, for example, heat=1 means that the clause that follows relies on hypotheses some of which are members of the hot list. The use of the hot list strategy often significantly enhances the power of a reasoning program. Finally, the ANSWER literal serves various purposes including tracking (by the researcher) the progress of the program's attack, containing upon completion of a constructive proof the object that is found; it plays no direct role in the program's reasoning.

Proof 1 of the Generalization of the Associativity of Logical ***or***

```
----> UNIT CONFLICT at 134.15 sec ----> 13467 [binary,13466.1,6.1]
$ANS(lemma_gen2_21a).

Length of proof is 43.  Level of proof is 27.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
5 [] P(i(i(n(x),n(y)),i(y,x))).
6 [] -P(i(i(i(a,i(b,c),c)),i(i(b,c),d)),i(i(i(i(a,b),b),c),d))) |
  $ANS(lemma_gen2_21a).
65 [] -P(i(x,y)) | -P(x) | P(y).
66 [] P(i(x,i(y,x))).
67 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
68 [] P(i(i(i(x,y),y),i(i(y,x),x))).
69 [] P(i(i(n(x),n(y)),i(y,x))).
```



```
-----------------------------------
83 [hyper,1,3,3] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
86 (heat=1) [hyper,65,67,83] P(i(i(i(i(x,y),z),u),
   i(i(i(i(y,v),i(x,v)),z),u))).
87 (heat=1) [hyper,65,83,67] P(i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))).
92 (heat=2) [hyper,65,87,66] P(i(i(i(i(x,y),z),i(i(i(u,x),y),z))).
121 [hyper,1,83,83] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
150 [hyper,1,92,3] P(i(i(i(x,y),z),i(i(z,u),i(y,u)))).
165 (heat=1) [hyper,65,150,69] P(i(i(i(x,y),z),i(n(x),z))).
166 (heat=1) [hyper,65,150,68] P(i(i(i(i(x,y),y),z),i(x,z))).
202 (heat=2) [hyper,65,67,165] P(i(i(i(n(x),y),z),i(i(i(x,u),y),z))).
207 (heat=2) [hyper,65,67,166] P(i(i(i(x,y),z),i(i(i(i(x,u),u),y),z))).
314 [hyper,1,83,166] P(i(i(x,i(y,z)),i(y,i(x,z)))).
321 [hyper,1,166,83] P(i(i(x,y),i(i(z,x),i(z,y)))).
329 (heat=1) [hyper,65,67,314] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
404 [hyper,1,202,5] P(i(i(i(x,y),n(z)),i(z,x))).
428 [hyper,1,207,166] P(i(i(i(i(i(i(x,y),y),z),z),u),i(x,u))).
445 [hyper,1,314,314] P(i(x,i(i(y,i(x,z)),i(y,z)))).
460 (heat=1) [hyper,65,445,66] P(i(i(x,i(i(y,i(z,y)),u)),i(x,u))).
525 [hyper,1,121,321] P(i(i(x,i(y,z)),i(i(z,u),i(x,i(y,u))))).
590 [hyper,1,329,121] P(i(i(x,i(y,z)),i(i(u,x),i(y,i(u,z))))).
601 (heat=1) [hyper,65,590,68] P(i(i(x,i(i(y,z),z)),i(i(z,y),i(x,y)))).
648 [hyper,1,87,428] P(i(i(i(i(i(i(i(i(x,y),y),z),z),u),v),w),
    i(i(i(x,u),v),w))).
668 [hyper,1,83,525] P(i(i(x,y),i(i(z,u),i(i(y,z),i(x,u))))).
680 (heat=1) [hyper,65,668,69] P(i(i(x,y),i(i(i(z,u),x),
    i(i(n(u),n(z)),y)))).
745 [hyper,1,525,601] P(i(i(i(x,y),z),i(i(x,i(i(y,u),u)),i(i(u,y),z)))).
746 [hyper,1,329,601] P(i(i(i(x,y),i(z,y)),i(i(y,x),i(z,x)))).
768 [hyper,1,745,404] P(i(i(i(x,y),i(i(n(z),u),u)),i(i(u,n(z)),i(z,x)))).
769 (heat=1) [hyper,65,768,69] P(i(i(x,n(y)),i(y,n(x)))).
804 [hyper,1,590,769] P(i(i(x,i(y,n(z))),i(z,i(x,n(y))))).
872 [hyper,1,202,804] P(i(i(i(x,y),i(z,n(u))),i(u,i(n(x),n(z))))).
933 [hyper,1,83,872] P(i(i(x,y),i(z,i(n(y),n(x))))).
13120 [hyper,1,460,933] P(i(i(x,y),i(n(y),n(x)))).
13131 [hyper,1,680,13120] P(i(i(i(x,y),i(z,u)),i(i(n(y),n(x)),
      i(n(u),n(z))))).
13132 [hyper,1,668,13120] P(i(i(x,y),i(i(i(n(z),n(u)),x),i(i(u,z),y)))).
13221 (heat=1) [hyper,65,13132,69] P(i(i(i(n(x),n(y)),i(n(z),n(u))),
      i(i(y,x),i(u,z)))).
13324 [hyper,1,321,13221] P(i(i(x,i(i(n(y),n(z)),i(n(u),n(v)))),
      i(x,i(i(z,y),i(v,u))))).
13398 [hyper,1,13324,746] P(i(i(i(n(x),n(y)),i(n(z),n(y))),i(i(x,y),
      i(x,z)))).
13399 [hyper,1,321,13398] P(i(i(x,i(i(n(y),n(z)),i(n(u),n(z)))),
      i(x,i(i(y,z),i(y,u))))).
13400 [hyper,1,13399,13131] P(i(i(i(x,y),i(x,z)),i(i(y,x),i(y,z)))).
13401 (heat=1) [hyper,65,67,13400] P(i(i(i(i(x,y),i(x,z)),u),
      i(i(i(y,x),i(y,z)),u))).
```

```
13460 [hyper,1,13401,83] P(i(i(i(i(x,y),i(z,y)),i(i(x,y),u)),
      i(i(x,z),i(i(z,y),u)))).
13461 [hyper,1,86,13460] P(i(i(i(i(i(x,y),z),i(i(u,y),z)),i(i(u,y),v)),
      i(i(u,x),i(i(x,y),v)))).
13464 [hyper,1,460,13461] P(i(i(i(i(i(i(x,y),z),u),i(i(y,z),u)),
      i(i(y,z),v)),i(i(i(x,y),z),v))).
13466 [hyper,1,648,13464] P(i(i(i(x,i(i(y,z),z)),i(i(y,z),u)),
      i(i(i(i(x,y),y),z),u))).
```

Proof 2 of the Generalization of the Associativity of Logical ***or***

```
----> UNIT CONFLICT at   0.68 sec ----> 1131 [binary,1130.1,9.1]
$ANS(lemma_gen2_21b).

Length of proof is 12.  Level of proof is 10.

---------------- 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)))).
9 [] -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).
25 [] -P(i(x,y)) | -P(x) | P(y).
27 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
28 [] P(i(i(i(x,y),y),i(i(y,x),x))).
------------------------------------
34 [hyper,1,3,3] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
36 [hyper,1,3,2] P(i(i(i(x,y),z),i(y,z))).
47 (heat=1) [hyper,25,36,28] P(i(x,i(i(x,y),y))).
70 [hyper,1,34,34] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
108 [hyper,1,70,47] P(i(i(x,i(y,z)),i(y,i(x,z)))).
125 (heat=1) [hyper,25,27,108] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
209 [hyper,1,125,70] P(i(i(x,i(y,z)),i(i(u,x),i(y,i(u,z))))).
235 (heat=1) [hyper,25,209,28] P(i(i(x,i(i(y,z),z)),i(i(z,y),i(x,y)))).
396 [hyper,1,3,235] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,i(i(y,x),x)),u))).
441 (heat=1) [hyper,25,396,27] P(i(i(x,i(i(y,z),z)),
    i(i(i(x,y),u),i(i(z,y),u)))).
737 [hyper,1,209,441] P(i(i(x,i(y,i(i(z,u),u))),i(i(i(y,z),v),
    i(x,i(i(u,z),v))))).
1130 [hyper,1,737,441] P(i(i(i(i(i(x,y),y),z),u),i(i(x,i(i(y,z),z)),
     i(i(y,z),u)))).
```

We now turn to the promised elegant proof of the dependence of the fifth of the Łukasiewicz axioms on the first four. That proof is also free of double negation, which brings us to yet another question of possible interest to logicians.

- QUESTION. Where P and Q may each be collections of formulas, if T is a theorem asserting the deducibility of Q from P such that Q is free of double negation, what conditions guarantee that there exists a proof relying solely on condensed detachment all of whose deduced steps are free of double negation?

Proof of the Dependence of MV5 on MV1 through MV4

```
----> UNIT CONFLICT at   4.59 sec ----> 5152 [binary,5151.1,20.1]
$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)))).
5 [] P(i(i(n(x),n(y)),i(y,x))).
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).
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))).
97 [hyper,1,3,5] P(i(i(i(x,y),z),i(i(n(y),n(x)),z))).
127 [hyper,1,40,40] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
183 [hyper,1,97,42] P(i(i(n(x),n(i(y,z))),i(z,x))).
185 (heat=1) [hyper,26,28,183] P(i(i(i(x,y),z),i(i(n(y),n(i(u,x))),z))).
259 [hyper,1,3,61] P(i(i(i(n(x),y),z),i(i(i(x,u),y),z))).
278 (heat=1) [hyper,26,259,30] P(i(i(i(x,y),n(z)),i(z,x))).
348 [hyper,1,127,49] P(i(i(x,i(y,z)),i(y,i(x,z)))).
357 (heat=1) [hyper,26,28,348] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
587 [hyper,1,357,127] P(i(i(x,i(y,z)),i(i(u,x),i(y,i(u,z))))).
600 (heat=1) [hyper,26,587,29] P(i(i(x,i(i(y,z),z)),i(i(z,y),i(x,y)))).
626 (heat=2) [hyper,26,600,30] P(i(i(x,y),i(i(n(x),n(i(y,x))),y))).
904 [hyper,1,587,278] P(i(i(x,i(i(y,z),n(u))),i(u,i(x,y)))).
974 [hyper,1,357,600] P(i(i(i(x,y),i(z,y)),i(i(y,x),i(z,x)))).
1942 [hyper,1,904,626] P(i(x,i(i(y,n(x)),n(y)))).
2311 [hyper,1,127,1942] P(i(i(x,i(y,n(z))),i(z,i(x,n(y))))).
2546 [hyper,1,259,2311] P(i(i(i(x,y),i(z,n(u))),i(u,i(n(x),n(z))))).
3139 [hyper,1,40,2546] P(i(i(x,y),i(z,i(n(y),n(x))))).
3303 [hyper,1,600,3139] P(i(i(i(n(x),n(y)),z),i(i(y,x),z))).
```

```
3304 [hyper,1,587,3139] P(i(i(x,i(y,z)),i(u,i(x,i(n(z),n(y))))))).
3672 [hyper,1,3304,49] P(i(x,i(y,i(n(z),n(i(y,z))))))).
3723 (heat=1) [hyper,26,3672,30] P(i(x,i(n(y),n(i(x,y))))).
4209 [hyper,1,185,3723] P(i(i(n(x),n(i(y,z))),i(n(u),n(i(i(z,x),u))))).
4622 [hyper,1,974,4209] P(i(i(n(i(i(x,y),x)),n(y)),i(n(x),n(y)))).
4843 [hyper,1,3303,4622] P(i(i(x,i(i(y,x),y)),i(n(y),n(x)))).
5027 [hyper,1,357,4843] P(i(i(i(x,y),i(y,x)),i(n(x),n(y)))).
5068 (heat=1) [hyper,26,28,5027] P(i(i(i(n(x),n(y)),z),
     i(i(i(x,y),i(y,x)),z))).
5151 (heat=2) [hyper,26,5068,30] P(i(i(i(x,y),i(y,x)),i(y,x))).
```

The given proof offers added interest in that, in contrast to, say, the proof found in Rose and Rosser, it does *not* rely on the use of frequently cited lemmas. For example, in the notation of Rose and Rosser, Lemmas 3.5 and 3.51 are not used. Whether Lemma 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.

In addition to the generalizations of associativity reported earlier, another notable success in the area of infinite-valued sentential calculus merits mention. Rose and Rosser [Rose1958, page 12] note that they are unable to prove several distributivity laws from the axioms of Łukasiewicz's intinite-valued sentential logic. It is easy to show that these distributivity laws are valid in the semantics of infinite-valued logic. And, since Rose and Rosser were able to prove the completeness of the axioms of infinite-valued logic, they knew that these distributive laws must be provable from the axioms. However, condensed detachment proofs of these laws from the axioms of infinite-valued logic eluded Rose and Rosser. Recently, Harris and Fitelson [Harris2000] have used OTTER to find condensed detachment proofs of these elusive distributivity laws.

4.2. Two-valued sentential calculus

The focus in this section is on two-valued sentential (or propositional) calculus, for which many axiom systems exist. This area of logic is stronger than is infinite-valued sentential calculus in the sense that the axioms for the latter are deducible from an axiom set of the former. Łukasiewicz [Łukasiewicz1963] provided the following axiom system, expressed in clause notation.

```
%  Lukasiewicz 1 2 3.
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))).
```


Łukasiewicz also provided a single axiom for this area of logic, the following expressed in clause notation.

```
%  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))))).
```

However, in the paper in which he offered this fine axiom he did not include a proof. Nor, from what we know, does the literature of logic offer a proof, which (in effect) leads to an open question. Specifically, find a proof showing that the given formula axiomatizes two-valued sentential calculus.

With our emphasis on axiomatic proofs, our objective was to find a proof that relies solely on condensed detachment, that begins with the single 23-letter formula, and that deduces some known axiom system. For our study, we added an additional constraint, one that is pertinent to a question we posed earlier for logicians. In particular, we sought a proof free of double negation — and we succeeded. Our proof completes by deducing the earlier-given three-axiom system of Łukasiewicz.

A Proof of the 23-Letter Łukasiewicz Single Axiom

```
-----> EMPTY CLAUSE at   2.02 sec ----> 123 [hyper,7,122,106,57]
$ANS(step_allLuka_1_2_3).

Length of proof is 56.  Level of proof is 43.

---------------- 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(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
----------------------------------
18 [hyper,1,8,8] P(i(x,i(i(i(i(y,z),i(u,z)),i(z,v)),i(w,i(z,v))))).
19 [hyper,1,18,18] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
20 [hyper,1,8,19] P(i(x,i(i(i(y,z),i(i(u,y),i(v,y))),
   i(w,i(i(u,y),i(v,y)))))).
21 [hyper,1,19,8] P(i(x,i(y,i(i(y,z),i(u,z))))).
22 [hyper,1,20,20] P(i(i(i(x,y),i(i(z,x),i(u,x))),i(v,i(i(z,x),i(u,x))))).
23 [hyper,1,21,21] P(i(x,i(i(x,y),i(z,y)))).
24 [hyper,1,22,23] P(i(x,i(i(i(y,z),y),i(u,y)))).
25 [hyper,1,24,24] P(i(i(i(x,y),x),i(z,x))).
28 [hyper,1,8,25] P(i(x,i(i(y,i(y,z)),i(u,i(y,z))))).
33 [hyper,1,28,28] P(i(i(x,i(x,y)),i(z,i(x,y)))).
38 [hyper,1,33,33] P(i(x,i(i(y,i(y,z)),i(y,z)))).
40 [hyper,1,38,38] P(i(i(x,i(x,y)),i(x,y))).
```

```
42 [hyper,1,40,8] P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(i(z,x),i(u,x)))).
43 [hyper,1,42,42] P(i(i(i(x,y),i(y,z)),i(u,i(y,z)))).
44 [hyper,1,42,40] P(i(i(x,i(i(n(x),n(y)),z)),i(y,i(i(n(x),n(y)),z)))).
46 [hyper,1,42,43] P(i(i(i(x,y),i(z,x)),i(u,i(z,x)))).
47 [hyper,1,40,43] P(i(i(i(x,y),i(y,z)),i(y,z))).
49 [hyper,1,40,46] P(i(i(i(x,y),i(z,x)),i(z,x))).
50 [hyper,1,47,8] P(i(i(i(i(n(x),n(y)),z),x),i(i(x,u),i(y,u)))).
51 [hyper,1,49,23] P(i(i(i(i(x,y),z),y),i(x,y))).
52 [hyper,1,50,47] P(i(i(i(n(x),y),z),i(x,z))).
54 [hyper,1,51,46] P(i(x,i(y,i(z,x)))).
57 [hyper,1,52,40] P(i(x,i(n(x),y))).
58 [hyper,1,52,54] P(i(x,i(y,i(z,i(n(x),u))))).
61 [hyper,1,47,58] P(i(x,i(y,i(n(i(z,x)),u)))).
64 [hyper,1,44,61] P(i(x,i(i(n(y),n(x)),i(n(i(z,y)),u)))).
66 [hyper,1,64,44] P(i(i(n(x),n(i(i(y,i(i(n(y),n(z)),u)),
   i(z,i(i(n(y),n(z)),u))))),i(n(i(v,x)),w))).
67 [hyper,1,49,66] P(i(n(i(x,y)),n(y))).
68 [hyper,1,23,67] P(i(i(i(n(i(x,y)),n(y)),z),i(u,z))).
70 [hyper,1,50,68] P(i(i(i(x,y),z),i(y,z))).
73 [hyper,1,70,8] P(i(i(i(i(n(x),n(y)),z),x),i(u,i(i(x,v),i(y,v))))).
76 [hyper,1,51,73] P(i(i(n(x),n(y)),i(z,i(i(x,u),i(y,u))))).
77 [hyper,1,42,73] P(i(i(i(i(x,y),i(z,y)),i(i(n(x),n(z)),u)),
   i(v,i(i(n(x),n(z)),u)))).
79 [hyper,1,42,76] P(i(i(i(i(x,y),i(z,y)),n(x)),i(u,n(x)))).
81 [hyper,1,77,33] P(i(x,i(i(n(y),n(i(y,z))),i(i(y,z),z)))).
83 [hyper,1,42,79] P(i(i(n(x),i(i(x,y),i(z,y))),i(u,i(i(x,y),i(z,y))))).
86 [hyper,1,81,81] P(i(i(n(x),n(i(x,y))),i(i(x,y),y))).
87 [hyper,1,40,83] P(i(i(n(x),i(i(x,y),i(z,y))),i(i(x,y),i(z,y)))).
88 [hyper,1,52,86] P(i(x,i(i(x,y),y))).
91 [hyper,1,23,88] P(i(i(i(x,i(i(x,y),y)),z),i(u,z))).
92 [hyper,1,77,91] P(i(x,i(i(n(y),n(z)),i(z,i(i(y,u),u))))).
97 [hyper,1,92,92] P(i(i(n(x),n(y)),i(y,i(i(x,z),z)))).
99 [hyper,1,52,97] P(i(x,i(y,i(i(x,z),z)))).
100 [hyper,1,87,99] P(i(i(x,y),i(i(n(x),y),y))).
102 [hyper,1,70,99] P(i(x,i(y,i(i(i(z,x),u),u)))).
103 [hyper,1,87,102] P(i(i(x,y),i(i(i(z,n(x)),y),y))).
105 [hyper,1,91,103] P(i(x,i(i(i(y,n(z)),i(i(z,u),u)),i(i(z,u),u)))).
106 [hyper,1,49,100] P(i(i(n(x),x),x)).
107 [hyper,1,42,105] P(i(i(i(i(x,y),y),z),i(x,z))).
108 [hyper,1,107,103] P(i(x,i(i(i(y,n(i(x,z))),z),z))).
109 [hyper,1,70,108] P(i(x,i(i(i(y,n(i(i(z,x),u))),u),u))).
113 [hyper,1,46,109] P(i(x,i(i(i(y,n(i(i(z,i(u,v)),u))),u),u))).
116 [hyper,1,42,113] P(i(i(x,y),i(i(i(z,i(x,u)),x),y))).
118 [hyper,1,116,108] P(i(i(i(x,i(y,z)),y),i(i(i(u,n(i(y,v))),v),v))).
120 [hyper,1,42,118] P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))).
122 [hyper,1,120,23] P(i(i(x,y),i(i(y,z),i(x,z)))).
123 [hyper,7,122,106,57] $ANS(step_allLuka_1_2_3).
```


The original approach we took (detailed in [Wos2000a]) also yielded deductions of Hilbert's and Church's axiom systems. At this point, we do not know whether a shorter proof (than 56 applications of condensed detachment) exists with the given constraints. Nor do we know whether a shorter proof exists if one removes the requirement of avoiding double negation. For the researcher who might find either question intriguing enough to merit study, we note that our first proof for the 23-letter single axiom has length 200. In view of this fact, one might (in an attempt to answer either question) be forced to formulate an attack that (in its first stages) yields a proof far longer than desired.

Regarding the concern for proof length and the desire for finding proofs shorter than that in hand, one need only turn to the work of Meredith and Prior, Thomas, Ulrich, and others. Indeed, Meredith and Prior [Meredith1963, page 171] present a "very slight abridgement" of Łukasiewicz's proof [Łukasiewicz1970, pages 299–300] of the sufficiency of his shortest single axiom for the implicational fragment of two-valued sentential (or propositional) calculus. Łukasiewicz's proof (when reasonably reconstructed from his detachment and substitution proof) requires 34 condensed detachment steps. Meredith and Prior were able to eliminate one step of Łukasiewicz's proof, yielding a 33-step condensed detachment proof, four of whose steps are not present in the reconstructed Łukasiewicz proof. Using OTTER, we have been able to find a 32-step proof (the following), two of whose steps are not present in the 33-step proof of Meredith and Prior, and four of whose steps are not present in the Łukasiewicz 34-step proof. One of the two just-mentioned steps is also absent from the Lukaseiwicz proof, and two of the four just-mentioned steps are among the four present in the 33-step proof but not in the 34-step proof.

A Proof of the Łukasiewicz Shortest Single Axiom

```
-----> EMPTY CLAUSE at   0.23 sec ----> 82 [hyper,34,53,77,79]
$ANS(TARSKI_BERNAYS).

Length of proof is 32.  Level of proof is 29.

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

33 [] -P(i(x,y)) | -P(x) | P(y).
34 [] -P(i(p,i(q,p))) | -P(i(i(i(p,q),p),p))
   | -P(i(i(p,q),i(i(q,r),i(p,r)))) |
   $ANS(TARSKI_BERNAYS).
35 [] P(i(i(i(x,y),z),i(i(z,x),i(u,x)))).
---------------------------------------
```

```
44 [hyper,33,35,35] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
45 [hyper,33,35,44] P(i(i(i(x,i(y,z)),i(i(u,y),i(v,y))),
   i(w,i(i(u,y),i(v,y))))).
46 [hyper,33,45,35] P(i(x,i(i(i(y,z),y),i(u,y)))).
47 [hyper,33,46,46] P(i(i(i(x,y),x),i(z,x))).
48 [hyper,33,35,47] P(i(i(i(x,y),i(y,z)),i(u,i(y,z)))).
49 [hyper,33,35,48] P(i(i(i(x,i(y,z)),i(u,y)),i(v,i(u,y)))).
50 [hyper,33,35,49] P(i(i(i(x,i(y,z)),i(u,i(z,v))),i(w,i(u,i(z,v))))).
51 [hyper,33,50,35] P(i(x,i(i(i(y,z),u),i(z,u)))).
52 [hyper,33,51,51] P(i(i(i(x,y),z),i(y,z))).
53 [hyper,33,52,52] P(i(x,i(y,x))).
55 [hyper,33,52,35] P(i(x,i(i(x,y),i(z,y)))).
56 [hyper,33,35,55] P(i(i(i(i(i(x,y),z),i(u,z)),x),i(v,x))).
57 [hyper,33,35,56] P(i(i(i(x,y),i(i(i(y,z),u),i(v,u))), i(w,i(i(i(y,z),u),
   i(v,u))))).
58 [hyper,33,35,57] P(i(i(i(x,i(i(i(y,z),u),i(v,u))),i(w,y)),i(v6,i(w,y)))).
59 [hyper,33,35,58] P(i(i(i(x,i(y,z)),i(u,i(i(i(z,v),w),i(v6,w)))),
   i(v7,i(u,i(i(i(z,v),w),i(v6,w)))))).
60 [hyper,33,59,35] P(i(x,i(i(i(y,z),i(u,v)),i(i(i(z,w),v),i(u,v))))).
61 [hyper,33,60,60] P(i(i(i(x,y),i(z,u)),i(i(i(y,v),u),i(z,u)))).
62 [hyper,33,61,35] P(i(i(i(x,y),i(z,u)),i(i(x,u),i(z,u)))).
63 [hyper,33,62,55] P(i(i(x,i(y,z)),i(i(i(x,u),z),i(y,z)))).
64 [hyper,33,63,35] P(i(i(i(i(i(x,y),z),u),i(v,x)),i(i(z,x),i(v,x)))).
65 [hyper,33,35,64] P(i(i(i(i(x,y),i(z,y)),i(i(i(y,u),x),v)),
   i(w,i(i(i(y,u),x),v)))).
66 [hyper,33,65,65] P(i(x,i(i(i(i(y,z),u),i(v,z)),i(i(i(z,w),v),i(y,z))))).
67 [hyper,33,66,66] P(i(i(i(i(i(x,y),z),i(u,y)),i(i(i(y,v),u),i(x,y)))).
68 [hyper,33,61,67] P(i(i(i(i(i(x,y),z),i(u,y)),i(i(i(y,v),x),i(u,y)))).
69 [hyper,33,67,68] P(i(i(i(i(i(x,y),z),i(i(y,u),v)),i(i(v,y),i(x,y)))).
70 [hyper,33,68,62] P(i(i(i(i(i(x,y),z),u),i(i(u,y),i(x,y)))).
71 [hyper,33,69,64] P(i(i(i(x,y),z),i(i(i(y,u),z),z))).
73 [hyper,33,64,71] P(i(i(x,y),i(i(i(x,z),y),y))).
75 [hyper,33,73,73] P(i(i(i(i(x,y),z),i(i(i(x,u),y),y)),i(i(i(x,u),y),y))).
76 [hyper,33,70,75] P(i(i(i(i(i(x,y),z),z),u),i(i(x,z),u))).
77 [hyper,33,75,71] P(i(i(i(x,y),x),x)).
79 [hyper,33,76,70] P(i(i(x,y),i(i(y,z),i(x,z)))).
```

4.3. Additional successes

Wajsberg [Wajsberg1977, Theorem 37, page 209] gives a generalization of the Tarski-Bernays axiomatization of the implicational fragment of two-valued sentential logic. Wajsberg's proof relies on mathematical induction. Using OTTER, we have found the following 17-step, pure condensed detachment proof of Wajsberg's theorem.

A Proof of Wajsberg's Theorem 37

```
-----> EMPTY CLAUSE at   0.05 sec ----> 148 [hyper,24,146,27,28]
$ANS(TARSKI_BERNAYS).

Length of proof is 17.  Level of proof is 9.

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

23 [] -P(i(x,y)) | -P(x) | P(y).
24 [] -P(i(p,i(q,p))) | -P(i(i(i(p,q),p),p))
   | -P(i(i(p,q),i(i(q,r),i(p,r)))) |
   $ANS(TARSKI_BERNAYS).
25 [] P(c1).
26 [] P(i(x,i(c1,x))).
27 [] P(i(i(i(x,y),x),x)).
28 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
----------------------------------
32 [hyper,23,28,28] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
34 [hyper,23,28,27] P(i(i(x,y),i(i(i(x,z),x),y))).
35 [hyper,23,28,26] P(i(i(i(c1,x),y),i(x,y))).
40 [hyper,23,32,27] P(i(i(x,i(x,y)),i(x,y))).
50 [hyper,23,32,35] P(i(i(x,c1),i(y,i(x,y)))).
53 [hyper,23,35,34] P(i(x,i(i(i(c1,y),c1),x))).
58 [hyper,23,32,40] P(i(i(i(x,y),x),i(i(x,y),y))).
70 [hyper,23,28,53] P(i(i(i(i(i(c1,x),c1),y),z),i(y,z))).
85 [hyper,23,32,70] P(i(i(x,i(i(c1,y),c1)),i(z,i(x,z)))).
89 [hyper,23,70,58] P(i(i(c1,x),i(i(i(c1,x),c1),c1))).
98 [hyper,23,32,85] P(i(i(i(c1,x),y),i(z,i(i(y,c1),z)))).
105 [hyper,23,35,89] P(i(x,i(i(i(c1,x),c1),c1))).
115 [hyper,23,27,98] P(i(c1,i(i(x,c1),c1))).
125 [hyper,23,28,105] P(i(i(i(i(i(c1,x),c1),c1),y),i(x,y))).
131 [hyper,23,115,25] P(i(i(x,c1),c1)).
141 [hyper,23,125,131] P(i(x,c1)).
146 [hyper,23,50,141] P(i(x,i(y,x))).
148 [hyper,24,146,27,28] $ANS(TARSKI_BERNAYS).
```

Meredith [Meredith1953] gives two 19-letter single axioms for the system ⟨C,O⟩ of two-valued propositional calculus. Meredith [Meredith1953, pages 160–163] reports a 31-step detachment plus substitution sufficiency proof for the first of these axioms. His proof is not a condensed detachment proof, relying on unneeded identifications of variables in several steps. Using OTTER, we have determined that the shortest condensed detachment proof that contains Meredith's 31 reported steps is in fact a 37-step, 8-variable condensed detachment proof. Using OTTER, we have found a 26-step, 6-variable condensed detachment proof. Our OTTER proof is significantly

more elegant than Meredith's, in terms of its length, and in terms of the complexity of the formulas that appear in the proof.

As for the second ⟨C,O⟩ single axiom, Meredith later reports a proof in [Meredith1963, pages 183–184]. This proof requires 60 condensed detachment steps. Using OTTER, we have found a 54-step condensed detachment proof.

Meredith and Prior [Meredith1963, page 182] report a single axiom for the two formulas P(i(i(i(x,x),y),y)) and P(i(i(x,y),i(i(y,z),i(x,z)))) (known in the modern literature as "specialized assertion" and "suffixing", respectively). Meredith gives a proof of the two desired formulas from his single axiom. But, Meredith's proof uses combinators and lambda conversion, not condensed detachment on propositional formulas. As such, an explicit condensed detachment proof was missing. Using OTTER, we have found a 7-step condensed detachment proof. Since we found this proof using exhaustive (breadth-first) search, it is likely that no shorter condensed detachment proof exists. We have also used OTTER to prove the other direction of the equivalence. We have a 6-step condensed detachment proof of Meredith's axiom from specialized assertion and suffixing.

Meredith [Meredith1953, pages 157–160] gives a 41-step condensed detachment proof of the sufficiency of his 21-letter single axiom for the ⟨C,N⟩ system of two-valued sentential logic. Meredith's proof uses 7 distinct variables and relies heavily upon the use of double-negation terms. Using OTTER, Deepak Kapur found a 6-variable, 63-step condensed detachment proof. We now have a 50-step, 6-variable proof (containing double-negation terms). We have also used OTTER to find a 51-step, 7-variable proof that is free of double negation terms. It remains open whether there exists a 6-variable proof that is also free of double-negation terms.

Meredith [Meredith1963, page 172] reports two 19-letter single axioms for the ⟨C,I⟩ system of the implicational fragment of two-valued sentential logic. He proves the sufficiency of the first of these, using 38 condensed detachment steps. Using OTTER, we have found a 31-step condensed detachment proof. Meredith reports no proof of the sufficiency of his second ⟨C,I⟩ single axiom. Using OTTER, we have found a 61-step condensed detachment sufficiency proof of this second ⟨C,I⟩ single axiom.

In contrast to the other calculi on which we have focused, equivalential calculus is not concerned with implication. Also of note, that area of logic does admit shortest single axioms. The sufficiency of the single axioms *XHN* [e(x,e(e(y,z),e(e(z,x),y)))] and *XHK* [e(x,e(e(y,z),e(e(x,z),y)))] for equivalential calculus, first proved by S. Winker in the early 1980s by in part relying on one of Argonne's automated reasoning programs, has also

been proved more recently using OTTER. Indeed, OTTER furnished very elegant proofs of the sufficiency of these two single axioms, a 19-step condensed detachment proof for *XHN* when compared with the Winker original 15-step proof, and a 23-step condensed detachment proof for *XHK* when compared with the Winker original 84-step proof. When *XHN* and *XHK* were dispatched, only one of the possible 630 formulas of length eleven (measured in symbol count) remained in doubt regarding its status in the context of being a shortest single axiom, namely, the formula *XCB* [e(x,e(e(e(x,y),e(z,y)),z))]. In that its status is still in doubt, we offer the following as a possible beginning. We have used OTTER to show that *XCB* implies reflexivity [e(x,x)], and that *XCB* plus symmetry [e(e(x,y),e(y,x))] is sufficient for equivalential calculus. The preceding reduces the open question to determining whether symmetry is deducible from *XCB*.

5. Summary and conclusions

Throughout many decades and in various areas of logic, axiomatic proofs of key theorems have continued to elude some of the finest minds. These missing proofs come in many forms. One class consists of results that are known to hold because of being valid, but no proof of any type is offered by the literature, much less an axiomatic proof. A second type consists of theorems whose only known proof is metatheoretic. A third type consists of announced results without proof (as in the case of the Łukasiewicz 23-letter single axiom). Various other types exist [Fitelson2000].

Then there are those proofs that are partially missing proofs because of being incomplete, not strictly missing, with steps left for the imagination. Closely related are those proofs that, although giving the impression that the standard inference rule (for example, condensed detachment) is used and no other, in fact contain steps not obtainable from that rule. These two classes of partially missing proofs, as well as those that are clearly types of (fully) missing proof, warrant effort aimed at producing the ideal proof, an axiomatic proof with all of the pertinent details.

For the spectrum of proofs to be found (including the cited five classes), OTTER has proved to be invaluable as an assistant. Not surprising are the successes at that end of the spectrum where a large fraction of the desired axiomatic proof is offered by the literature. However, where no clue exists concerning how to proceed or which target merits emphasis (as was the case with the Łukasiewicz 23-letter single axiom), successes indeed produce great satisfaction. Our successes with proof finding offer an additional (to us) startling property. In almost all cases, contrary to intuition, the proofs we

have found avoid reliance on double negation — absent are terms of the form $n(n(t))$ for any term t, where n denotes negation.

Motivated by the desire to produce a fruitful dialogue, we raise two questions. First, what conditions guarantee that there exists a proof relying solely on condensed detachment all of whose deduced steps are free of double negation? Second (and less precise), what is implied about the (apparently) implicitly held view concerning the necessity (in various areas of logic) of requiring the use of double negation? In addition to answers to the two posed questions, we ask for open questions to study, questions of the type featured in this article. For those with added curiosity in the context of automated reasoning, two books [Wos1999,Wos2000b] may serve one well; see Section 1.2 for information on the nature of the two cited books.

References

[Fitelson2000] Fitelson, B., and L. Wos, 'Missing proofs found', preprint ANL/MCS-P816-0500, Mathematics and Computer Science Division, Argonne National Laboratory, May 2000.

[Harris2000] Harris, K., and B. Fitelson, 'Distributivity in Lw and other sentential logics', preprint, 2000.

[Kalman1983] Kalman, J., 'Condensed detachment as a rule of inference', *Studia Logica* 42 (1983), 443–451.

[Łukasiewicz1963] Łukasiewicz, J., *Elements of Mathematical Logic*, Macmillan, New York, 1963.

[Łukasiewicz1970] Łukasiewicz, J., *Selected Works*, edited by L. Borkowski, North Holland, Amsterdam, 1970.

[McCune1994] McCune, W., *OTTER 3.0 Reference Manual and Guide*, Tech. Report ANL-94/6, Argonne National Laboratory, Argonne, IL, January 1994.

[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 Łukasiewicz', *Trans. AMS* 87, no. 1 (1958), 54.

[Meredith1963] Meredith, C. A., and A. Prior, 'Notes on the axiomatics of the propositional calculus', *Notre Dame J. Formal Logic* 4, no. 3 (1963), 171–187.

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

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

[Wos1999] Wos, L., and G, W. Pieper, *A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning*, Singapore, World Scientific, 1999.

[Wos2000a] Wos, L., 'Conquering the Meredith single axiom', preprint ANL/MCS-P815-0500, Mathematics and Computer Science Division, Argonne National Laboratory, May 2000.

[Wos2000b] Wos, L., and G, W. Pieper, *The Collected Works of Larry Wos*, Singapore, World Scientific, 2000.

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

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