book7/ch02
created 11-30-2000
revised last 04-26-2001

CHAPTER 2

Methodology, Strategy, and a Significant Proof

delim @@

When one enters the arena of chess, poker, bridge, puzzle solving, and the like, great is the temptation to seek one or more algorithms that guarantee victory. Paradoxically, if such algorithms existed, then soon the fascination of the activity would disappear. In the place of algorithms, however, methodologies do exist based on diverse strategies; see the sidebar on Methodologies near the end of this chapter. The use of these methodologies markedly increases the likelihood of success.

Similar remarks hold for the pursuit of results in logic and in mathematics, especially when the seeker is joined by an automated reasoning program. Indeed, although many recognize that fine minds do not, for example, produce proofs by applying an algorithm, some expect that the situation should be different when a program is involved. For those with such an expectation, we note that (in our view) logic and mathematics are simply too deep to permit a practical algorithmic approach (which is part of their fascination). Nevertheless, as the result of years of experimentation, we have found effective methodologies, each relying on various strategies. Some of the methodologies are oriented toward finding a first proof or settling a conjecture, some oriented toward finding a better proof (for example, shorter or less complex), and some oriented toward both.

In this chapter, we begin our detailed discussion of methodology and the use of strategy. To illustrate the elements and the power, we focus on a proof missing from a 1936 paper by Lukasiewicz; see Section 2.5 for a satisfying result. In that paper he offers a remarkable formula (to be given shortly) consisting of twenty-three letters, a formula that provides a complete axiomatization for two-valued sentential (or propositional) calculus. Of course, one might be tempted to view problems in propositional logic as easy to solve and even trivial. Taking this view would be a mistake, for the following reasons. First, although a truth-table analysis will (in principle) determine whether a formula is true under all assignments of true and false, the analysis may be costly. This problem is encountered in hardware verification in industry, even for the design of integer multipliers. The number of lines in a truth table is exponential in the size of the formula. Second, although a truth-table analysis is useful in determining whether a given formula is a theorem (tautology) of propositional logic, it is of little or no use in determining whether a given formula or set of formulas provides an axiomatization. (A given formula is a theorem if and only if it evaluates to **true** under all assignments of **true** and **false**.) Indeed, when the question concerns the *deducibility* of one formula from another, a truth-table analysis is not useful, and, instead, inference rules come into play. Third, possessing a proof that a given formula has some property (such as being a single axiom or expressing the desired behavior of a circuit), we may wish to find a shorter proof. Here again truth tables provide no help. Especially the second and third aspects are not easily addressed, are often deep, and are of great interest in themselves; of even greater interest are the methodologies that can provide assistance in coping with either aspect.

In the Lukasiewicz 23-letter formula (the following), the function *i* denotes implication, and the function *n* denotes negation; the variables *x, y, z, u,* and *v* are universally quantified and mean ‘‘for all’’.

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

Because the Lukasiewicz result is so remarkable, surely a turn of the page will reveal his proof. If not, then turning a few more pages of his paper certainly will. But, unfortunately, such is not the case; we are denied insight into the Lukasiewicz mind. Indeed, one finds that the desired proof is *missing*, not included. Nor does a search of the literature of the next more than six decades unearth the desired proof. Our quarry (on May 13, 1999) was just such a proof.

The capture was made (on May 13) in less than four CPU-hours (on a Pentium II computer, 400 MHz) with four experiments with OTTER, each experiment building on the results of the preceding. ‘‘Experiments’’ is the correct term, rather than computer run, for we were testing a new methodology with no estimate of the likelihood of completing the desired proof for the Lukasiewicz 23-letter formula. Not counting the axiom under study, the completed proof has length 200 (applications of condensed detachment, which will be rigorously defined in Section 2.5). (That proof is found on the included CD-ROM.) In Sections 2.1, 2.2, and 2.3, we focus on the methodology that enabled OTTER to discover (from what we know) the first proof for the Lukasiewicz 23-letter single axiom, a proof missing for more than six decades.

Following the tradition of masters that include Meredith, Prior, and Thomas, we immediately began a search for an abridgment of the cited 200-step proof, seeking a proof with fewer (and perhaps far fewer) steps. Over the next few days, we continue to find shorter and still shorter proofs, eventually finding a 71-step proof. Then, upon revisiting the theorem almost a year later we found a most satisfying proof. Near the end of this chapter, we give that proof and touch on some of the means that we employ for finding shorter proofs. (Far more detail on proof shortening and proof refinement in other respects is given in a later chapter.) Since we know of no shorter proof, one has here (in effect) an open question for study.

Open question: Does there exist a proof shorter than length 56 relying solely on condensed detachment?

The various methodologies featured in this book depend on a number of elements that include options, parameters, and strategies. Several elements (as will be seen) are not naturally in the repertoire of the unaided researcher, nor are many offered in programs other than OTTER. Their availability (it seems to us) accounts in an important way for the successes featured in this volume. For but one (to-be-detailed) example—and here we come to a vital component of our various attacks—within OTTER's arsenal is the means to prevent retention of any conclusion in which a term of the form $n(n(t))$ occurs, where t is any term and n denotes negation. The literature suggests that such avoidance is contrary to the approach taken by logicians. On the other hand, the program's arsenal does not offer various powerful weapons employed by the researcher—knowledge, experience, intuition, and the ability to make wise judgments about paths of reasoning to follow. The key, then, is to skillfully combine the assets of a person's mind with the assets of an appropriate reasoning program so that far more than the sum results. The results of such teamwork are found in this volume, one proof after another that perhaps neither member of the team could have completed alone.

The 200-step proof of the beautiful Lukasiewicz result was obtained by applying one of various methodologies this book offers. In the remainder of this chapter, one is introduced to the key elements (some in fine detail) of this methodology, various obstacles that could confront a reasoning program, and the desired proof for the Lukasiewicz 23-letter formula. The first of the elements proved to be crucial (if judged by many later experiments with other theorems), an element somewhat reminiscent of the knowledge accrual that occurs in much of unaided research.

2.1. Lemma Adjunction

When a paper in logic or mathematics presents in detail the proof of a significant result, the path that culminates in the result is sprinkled with lesser results or lemmas. The development of a theory also exhibits such sprinkling. For example, in group theory, one might encounter the proof of the lemma that asserts that the inverse of the inverse of x is x. In two-valued sentential (or propositional) calculus, the development might include proofs of the lemmas that respectively show that $n(n(x))$ implies x and conversely, where n denotes negation. The correct expectation is that an included lemma will be relied upon later in the development. Indeed, from a slightly different viewpoint, lemmas are not proved and included on the *chance* that they might be needed. Of course, the research that precedes the paper or book might include the proof of lemmas that are discarded before publication by the author. The greater the master of a field under study, the smaller the likelihood of time and effort wasted on the proof of lemmas not put to good use.

Because an automated reasoning program is not expected to exhibit mastery of a field, its attack on a problem is strewn with deduced conclusions that play no role in the final result. Some of those conclusions merit the label of lemma, being of interest in themselves or in the context of proving a result other than that under study. In contrast to the unaided researcher, the program can afford to accrue hundreds of thousands of conclusions, even though the path from the hypotheses to the desired result utilizes a rather small number of deductions.

If the researcher suggested lemmas that might serve well, and if the program proved some or all of those lemmas, and if the researcher were correct, then the likelihood of success would be markedly increased. Although perhaps far from obvious, the desired proof might be found even when the smallest number of the suggested lemmas plays a role. The accuracy of this observation turns out to be crucial. Indeed, were the observation false, then what we next describe as ‘‘lemma adjunction’’ would most likely not be of much use.

The use of lemma adjunction addresses, sometimes vaguely and sometimes directly, the obstacle of the program’s lack of knowledge, experience, and intuition. It works in the following way; an example taken from our success with the Lukasiewicz 23-letter theorem will illustrate the various phases of this iterative component of the methodology under discussion. One begins (in the first phase) by selecting a set of lemmas for the program to attempt to prove. The lemmas need not be taken from the field of study from which the theorem under attack is taken. Each lemma must be representable by a single formula or equation, by a *unit clause* (a nonempty clause free of logical **or**). For example, the lemma ‘‘x implies x’’ is acceptable, but the lemma ‘‘if in a group $xy = y$, then $x = 1$, the identity element’’ is not acceptable. One need not be certain or, for that matter, even be much inclined to believe that the selected lemmas will be employed in a successful proof completion. Of course, if one is a master, then the lemma selection can reflect knowledge, experience, and intuition.

In the second phase of lemma adjunction, one takes the negation of each selected lemma—a clause that amounts to assuming the lemma is not provable—and place each negation in the passive list. Clauses in the passive list play no direct role in the program’s reasoning; they are used to detect proof completion, and they are used to purge newly deduced conclusions that are (in effect) redundant. Regarding provability, one need not choose lemmas that are clearly provable, and, one should expect that some of them will not be proved on the first try or perhaps ever proved. As will be seen, in the third phase, the inclusion of a lemma that one is certain cannot be proved can still be profitable. And here we encounter another difference between the attack a person might take and that which the program takes (when using lemma adjunction). Indeed, one would hardly waste a second in the attempt to prove a lemma conjectured to be unprovable, nor would one ask a colleague to waste time in this context.

In the third phase, for each selected lemma, one places its positive form in a list used by the program to direct the program’s attack. One also assigns a *weight* (priority) to reflect a conjectured importance of the lemma; the conjecture can be very weak. Technically, one relies in this phase on the *resonance strategy*, as the following illustrates.

weight(P(i(x,x)),2).

If and when the program proves the lemma x implies x, almost immediately the lemma will be chosen to drive the program’s reasoning. The lower the assigned value to a *resonator*, the higher priority it is given for reasoning direction.

As noted, even if an included lemma is not provable, its presence as a resonator can have a marked effect on the program’s attack because all variables in a resonator are treated as indistinguishable. Therefore, if the program deduces and retains a new conclusion (clause) A that is similar to an included resonator B, although A is not identical to B, the newly retained conclusion will be given the same priority as the lemma that prompted the inclusion of the resonator B. Clearly, an illustration is in order. Consider the following resonator B, and assume that B corresponds to a lemma chosen in phase one of lemma adjunction but not proved. (The clause language relies on predicates such as P, which here can be interpreted as ‘‘provable’’.)

weight(P(i(i(i(i(x,y),i(z,y)),y),i(i(z,x),z))),2).

If the program deduces and retains the following clause A, then A will be given high priority for directing the program's reasoning because (treating all variables as indistinguishable) *A* is similar to *B*.

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

The fact that *A* does not correspond to a chosen lemma but, nevertheless, is given high priority illustrates how the inclusion of an unprovable or unproved lemma can affect the program's reasoning. As for a profitable effect, a focus on *A* may be the key to completing a desired proof. In other words, (as illustrated) one need not choose the precise lemmas that play a vital role.

Again one sees a divergence between a person's approach and that taken by a reasoning program such as OTTER. It seems likely that a person focuses on lemmas that are appealing, rather than on the equivalence classes of which they are members. In this case, the equivalence class of a chosen lemma is the set of formulas or equations that are similar to it (in the sense of the resonance strategy), where all variables are treated as indistinguishable.

If, after applying the first three phases, OTTER finds the sought-after proof, keep in mind that the chosen lemmas play no role, for their inclusion is ignored in the program's reasoning. If, however, OTTER fails to find a proof—the typical case—the fourth phase is invoked. In this fourth phase, one prepares the way for another computer run by including in list(sos) the formulas or equations that are the clause equivalent of one of the chosen lemmas and that were proved in the first run. OTTER uses list(sos) to drive the program's attack. To ensure that all of its members are chosen to initiate the application of the included inference rules before any deduced clause is chosen, one includes the command set(sos_first). The main objective of this second run is to compete the desired proof; failing that, the secondary objective is to prove some of the chosen lemmas that were not proved in the first run.

If the second run does not produce the sought-after result, then one iterates with an ever-increasing list(sos), richer because of the inclusion of proved lemmas. The planned outcome has the program succeed after but a few runs. In the case of the Lukasiewicz 23-letter formula, four runs sufficed.

Before giving some of the pertinent details to the success with Lukasiewicz and discussing some of the variations of the just-discussed component of the methodology, we briefly focus on the promised illustration of lemma adjunction and then turn to another key component of the methodology. For our illustration, we focus on excerpts from the study that led to the formulation of the methodology under scrutiny. The study concerned the following Meredith single axiom (expressed in OTTER notation) for all of two-valued sentential calculus.

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

The objective was to find a means for OTTER to discover, essentially on its own, a *fully automated* proof that the given formula serves as a complete axiomatization. Among the targets used was the following three-axiom system of Lukasiewicz for that area of logic.

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))).

If OTTER could deduce the three Lukasiewicz axioms, then the program would succeed in showing that the Meredith axiom suffices. We knew of Meredith's proof, but the nature of the study precluded using any of that knowledge; to do otherwise would have been cheating. Many approaches were taken for approximately eight years before lemma adjunction was introduced by us. Its use, together with other components to be discussed in Sections 2.2 and 2.3, finally enabled the program to capture our quarry and *without* relying on Meredith's proof.

For phase 1, we chose various lemmas, including 68 (called theses by Lukasiewicz), for OTTER to attempt to prove. We had no defendable reason for making that choice, simply assuming that such lemmas must be important in some contexts because Lukasiewicz offered them. (Although we did not pause to make a comparison, we note that only 3 of the 68 chosen lemmas are present in the Meredith 41-step proof, and none of the 3 are members of the Lukasiewicz three-axiom system.)

For the second phase, we formed the negation of each of the 68, placing each in list(passive) with an ANSWER literal to enable one to know which was proved. For example, the following three clauses, corresponding respectively to theses 16, 17, and 8, were used.

```
-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).
```

For phase 3, we included resonators corresponding to each of the 68 chosen lemmas to cause OTTER to focus on a chosen lemma if and when it was proved. For example, we included the following three resonators.

```
weight(P(i(x,x)),2).
weight(P(i(x,i(i(y,x),x))),2).
weight(P(i(x,i(y,x))),2).
```

Because OTTER did not succeed in deriving the three-axiom system of Lukasiewicz or any of the other axiom systems we included as targets, the fourth phase came into play. Therefore, from among the chosen lemmas, we added to the (initial) set of support for a second run those proved in the first run. We continued in this manner and, in the fourth run, were rewarded with various proofs of interest. OTTER derived the desired Lukasiewicz axiom system, Church's, Frege's, an alternate of Lukasiewicz, and that of Wos. In summary, a new methodology was born, many of whose details will be covered in the remainder of this chapter.

The proof of the Lukasiewicz three-axiom system has length 160 (applications of condensed detachment) and level 74, where the level of an input hypothesis is 0 and that of a deduced conclusion is 1 greater than the maximum of the levels of its parents. The level of Meredith's 41-step proof is 30. (In Chapter 3, we return to an intense study of the Meredith single axiom.) Since we strongly suspect that an unaided researcher naturally seeks proofs of lower level, in contrast to a reasoning program having no preference in this regard, a proof of the type just cited might not be found without assistance from a program such as OTTER. Meredith's 41-step proof and OTTER's 160-step proof share 21 steps in common. As for the role the 68 chosen lemmas played, the following sheds some light.

Of the 160 steps, only 5 are among the 68 theses keyed upon, and only 10 of the 160 are similar to one of the 68 resonators. What we had shown with the Meredith study was the ability to find a missing proof. In other words, if Meredith had supplied no proof for his beautiful single axiom, with OTTER the proof could have been found.

As for the Lukasiewicz 23-letter single axiom, (as noted) no proof was available to us. We knew nothing about his choice of target or his approach to completing a proof. We did know that condensed detachment had not yet entered the picture in logic. Our attack on the theorem was motivated in part by the demand of testing the new methodology and in part by eagerness to find the missing proof. That attack, as noted, yielded the cited 200-step proof.

2.2. Term Avoidance

A second key component of the methodology that led to the discovery of a proof for the Lukasiewicz 23-letter single axiom concerns the avoidance of a type of term, namely, that in which double negation occurs. Such avoidance (which is an application of the *term avoidance strategy*) played a key role in our reported success with the Meredith single axiom. In Meredith's 41-step proof, 17 of the steps contain a term of the form $n(n(t))$ for some term t; in our 160-step proof, however, no such term is present.

The natural question concerns the motivation for our decision to avoid double negation. (By answering that question in some detail, one learns how our research interweaves various successes and often revisits earlier work.) The wellspring was mere curiosity. The time was the early 1990s; the general context was infinite-valued sentential calculus, an area of logic weaker than propositional calculus; the specific focus was a theorem (proved by Meredith) asserting the existence of a dependency among the five axioms Lukasiewicz offered for this area of logic. Meredith proved that what we term the fifth of these axioms is dependent on the other four. Our interest in this theorem had two phases. In the first, we had the objective of obtaining with OTTER a proof of the theorem relying solely on condensed detachment. In the second, our objective was to find a means for OTTER to prove the theorem *without* reliance on knowledge derived from an existing proof. In other words, that study shared with the cited Meredith study the objective of formulating an approach that enables a reasoning program to find what might be termed a fully automated proof. When such an objective is reached, quite often a new and useful and general approach is found, which is one of the reasons for conducting the corresponding study. Just as our reaching the objective for the Meredith single axiom resulted in a new and general methodology, our success with this theorem from infinite-valued sentential calculus led to a general approach.

As for the cited curiosity, the following piquant question arose (after we had found a means to obtain a fully automated proof). Can one find a proof of the dependence of the fifth of Lukasiewicz's five axioms for infinite-valued sentential calculus such that double-negation terms are absent? As part of the piquancy, a respected colleague hazarded that such was not possible, and a search of the relevant literature indicated that such a term-avoidance proof was not already in hand.

After various experiments, we found the desired proof (of the dependence of the fifth axiom), a proof of length 39. With more experimentation, eventually a 32-step proof avoiding double negation was completed, a proof given later in this book. The 32-step proof has added interest in that it is shorter than any offered by the earlier literature and in that it does not rely on two frequently used lemmas.

That success, although occurring a number of years before our attack on the cited Meredith problem focusing on his single axiom, was the wellspring for the inclusion of term avoidance in the methodology under discussion. Without this component, we strongly suspect that the 160-step proof for the Meredith single axiom as well as the earlier-cited 200-step for the Lukasiewicz 23-letter single axiom would have eluded us.

OTTER easily avoids double negation if one includes the following rewrite rule (*demodulator*) along with others that complete the task.

```
(n(n(x)) = junk).
```

When a conclusion is drawn in which double negation is present, with the given demodulator and others that complete the task, the conclusion is rewritten in such a manner that it is useless. Especially when equality is involved, one is advised to include something like the following to ensure that nonsense proofs are not produced.

```
weight(junk,1000).
```

Again one sees a sharp difference between a person's attack and that taken here. Indeed, it seems more than reasonable that the blocking of a large class of terms presents an obstacle to a person and is a counterintuitive move. For example, the literature often relies on lemmas that (in effect) assert the equivalence of x and $n(n(x))$. Such lemmas, even if deduced during the use of the given methodology, would immediately be purged. Further, it seems that, rather than hindering, the avoidance of a large class of terms markedly contributes to the likelihood of success for the reasoning program, perhaps because the density of useful information is sharply increased.

2.3. The Role of Prior Knowledge

As a reasoning program attacks a question or problem, it must ‘‘decide’’ where next to focus its attention, how to choose an item of information to initiate applications of the inference rules in use. Two obvious approaches exist. The program can rely on first-retained first-chosen (or breadth-first) search. Instead, the program can base its choice on the complexity of the items of information (clauses), typically choosing as the focus of attention the least complex available, the clause with the fewest symbols. In either case, syntax (or meaning) plays no role. After all, the number of symbols in a clause is only weakly correlated with significance.

In contrast to the preceding two approaches, a person (especially a master) can direct an attack based on knowledge, experience, and intuition. Indeed, even one who is new to a field will often have some intuition about answering a question or solving a problem. Therefore, the program seems handicapped in its attempt to complete an assignment. However, the resonance strategy can be employed to direct the reasoning based on prior knowledge. A taste of this strategy was provided in Section 2.1 in the context of lemma adjunction, where resonators were included that correspond to lemmas chosen for the program to attempt to prove. On the other hand, the inclusion of a resonator does not require its correspondent to be included as a possible lemma to be proved.

For example, consider the assignment of proving all three members of the three-axiom system of Lukasiewicz. Proofs of this type sometimes use the deduction of one of the members to aid in deducing another member. One can easily imagine using B in a proof of A when the objective is to prove A, B, and C Less obvious, but sometimes occurring, one can imagine using a conclusion that is similar to D to prove A, where the similarity rests with the ability to obtain B from D by replacing some or all variables by other variables. In other words, D and B are in the same equivalence class, being identical if one considers their variables as indistinguishable. Technically, D and B are captured by the corresponding resonator, a formula or equation whose variables are (in effect) ignored such that a value (weight) is assigned to it that reflects priority. In the example in focus, one could include as resonators the correspondents of each of the three Lukasiewicz axioms.

Indeed, part of the methodology under discussion in this chapter relies on the inclusion of resonators that correspond to the elements of the target. Further, the methodology encourages the inclusion of other resonators, some that are in no way connected to lemmas to be proved or quarry to be captured. One might, as we often do, include resonators that correspond to proof steps from a study only distantly related to that in focus. One can then assign a common value to all members of a given class of included resonators and, to the members of a different class, assign a different value. For example, one could assign the value 1 to resonators corresponding to members of one of the target axiom systems, 2 to the resonators corresponding to lemmas to be proved, and 3 to resonators corresponding to proof steps from another study. Including elements of the third type is indeed reminiscent of relying on prior knowledge.

This component of the methodology permits one to direct the program's attack based on preference, which in turn may be based on various factors. Although a program such as OTTER does not utilize a large (or small) database of previously proved results, definitions, and proofs—as a person does, especially one putting to excellent use years of study—employment of the resonance strategy shares in part the spirit of such reliance.

Along these lines, Ross Overbeek suggested that a study in which an appropriate reasoning program was participating should from its beginning have the researcher accrue sets of resonators. The sets might be taken from proofs the program finds, they might be taken from proofs found in the literature, or they might be simply supplied by the researcher without justification. In our research (especially as reflected in this volume), we do ''save'' sets of resonators from diverse successes with OTTER, choosing from among those sets based on conjectured relevance and on conjectured increase in program performance.

For a small taste of what will come in the context of finding a type of missing proof that focuses on proof length, and as evidence of the value of the methodology in focus, we note that the component currently under discussion is used in perhaps an unexpected way. Specifically, we often include in a later run as resonators items corresponding to the proof steps of a proof completed in an earlier run. In no way, however, do we consider this an example of ''program learning''. The facility of learning must wait for some eager devotee to undertake the difficult task of enhancing OTTER or some such program with this capacity.

2.4. Proof Contraction

Before turning to the prize, short proofs for the Lukasiewicz 23-letter single axiom, we pause to focus on the following easily-overlooked property of the proofs yielded by a successful use of the methodology highlighted in the three preceding sections. Specifically, when more than one run is required with the use of the methodology—because of the lemma-adjunction phase—the run that deduces the last remaining member of an axiom system generally relies on lemmas proved earlier. For example, where the goal was the deduction of the three-axiom system of Lukasiewicz from his 23-letter formula, the second and third members were deduced in the first run, but the first member was forced to wait until the third run. One finds as hypotheses in its proof, in addition to the use of the 23-letter formula, fourteen lemmas proved in the first two runs. Of course, in no way does that fact detract from the soundness (accuracy) of the proof.

In fact, the result is rather reminiscent of that which occurs in published papers in logic and in mathematics. In particular, if one wished to present or publish the proof in hand at the completion of the third run, one would first prove each of the fourteen lemmas, which in turn might require the proof of lemmas used in their proofs. An examination of the OTTER successes in the first two runs would yield the necessary information. In other words, as is true of so many papers, one would begin by proving some lemmas (perhaps none) that would then be used to prove fourteen additional lemmas. Then one would turn to the main theorem, rely on the already cited-and-proved lemmas, and (possibly) include a few additional proof steps and offer the final result. Indeed, many fine proofs take the form of citing earlier lemmas (such as 2.3, 3.22, 3.51, and the like) and finish with a few formulas or equations. The entire proof, including that required for the needed lemmas, may span ten or more pages.

Although one could with impunity take OTTER's successes with the given methodology, of course ignoring the lemmas proved that play no role in the final result, and produce a more-than-acceptable paper (in the spirit of the published literature), a far more attractive possibility exists. Indeed, with the goal of producing a proof with OTTER that displays just the hypothesis used to initiate the study—the goal of *proof contraction*—one could make one more run (as we did with our study of the 23-letter formula) in which all proof steps of all proofs (completed in the runs required by the methodology) are included as resonators. (Proof contraction is relevant only in the case in which lemma adjunction has played a role.) Although one might worry about introducing the use of lemmas that were not used in the final result, this move does avoid the necessity of recursively tracing back through the runs to retrieve just the proof steps relevant to the so-called extra lemmas cited in the final result. Of course, steps that occur in more than one proof of a used lemma are eliminated, which in part explains the use of the term ‘‘contraction’’.

Proof contraction can have, and often does have, most unexpected consequences. Indeed, proof contraction can discover a proof that obviates the need for many of the lemmas used in the final stage of the methodology. In the case of our study of the 23-letter formula, eight of the fourteen cited lemmas did not appear in the completed 200-step proof. If one totaled the number of steps in the proof being contracted, including those for the cited lemmas, but excluding duplicates, one would have a proof far longer than necessary. When (as is true for us in most of our research) the goal is to present a ‘‘short’’ proof, the likelihood of success is markedly increased if one begins with a proof of length k rather than length n where k is substantially less than n.

To find the desired contraction might be arduous by hand, especially if one has spent many months or years producing the sought-after result. However, to use OTTER to contract a proof of interest, which is our standard approach when lemma adjunction has been used, is essentially mechanical in nature. The procedure consists of (1) taking all of the proof steps of all of the lemmas that were used and all of the additional proof steps used to complete the theorem's proof, (2) removing all duplicate steps by sorting, (3) present the theorem in question to OTTER with the shorted proof steps as resonators each with the same small value assigned to it, (4) relying on a max_weight (maximum bound on the complexity of retained information) not much larger than the value assigned to the cited resonators, and (5) letting OTTER attack (almost certainly including the use of *ancestor subsumption* to aid in finding shorter proofs by comparing derivation lengths to the same conclusion). For the research of others, we conjecture that application of the approach just described could result in access to contracted proofs, proofs requiring far fewer pages. The program's assistance in this regard might be most profitable and, if successful, detract in no way from the original unaided (or aided) research.

2.5. A Short Proof for the Lukasiewicz 23-Letter Formula

At this point, we present the latest and shortest proof known to us that establishes the Lukasiewicz 23-letter formula to be a single axiom for two-valued sentential calculus. The proof completes with the deduction of the Lukasiewicz three-axiom system given earlier. As noted, our first success in this regard produced a 200-step proof (found on the included CD-ROM). That proof was obtained by applying the methodology whose key components were presented in the preceding three sections. Of course, the full story is not contained in the highlighted components. Rather, it is told by the input files (found on the CD-ROM) for the four experiments that enabled OTTER to find the 200-step proof.

The proof we give in this section was obtained by having OTTER apply various techniques (detailed in a later chapter) we formulated to discover shorter and ever-shorter proofs, often starting with a known proof. Our starting point was the cited 200-step proof; our (possibly temporary) stopping point is the following 56-step proof.

Regarding notation, as required by the clause language, each formula (whether appearing alone or in a disjunction) is preceded by a predicate symbol, in this case the predicate P interpretable as ‘‘provable’’. Logical **or** is denoted by ‘‘ | ’’ and logical **not** by ‘‘-’’. The function i denotes implication, and the function n denotes negation. The number of a clause reflects its position among those that are retained. The inference rule *condensed detachment* is captured by clause (1) in the following proof together with the use of the inference rule *hyperresolution*. Formally, condensed detachment (in the context of implication) 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. For example, if condensed detachment is applied to

i(x,i(x,i(y,y))) and
i(z,z)

with the second formula playing the role of C (the minor premiss), the following is obtained.

i(i(z,z),i(y,y)).

If the roles of the two formulas are reversed and condensed detachment is applied, a copy of the first formula is obtained. The designation [hyper,x,y,z] says that the nucleus for hyperresolution is clause (x), the major premiss is clause (y), and the minor premiss is clause (z). (This aspect of the notation is consistent with the papers in which condensed detachment is employed, namely, the y,z pair denotes the major premiss followed by the minor.) As one sees, clause (7) corresponds to assuming that the three-axiom system of Lukasiewicz is not deducible.

A Short Proof for the Lukasiewicz 23-*letter Formula*

```
----- Otter 3.0.5b, March 1998 -----
The process was started by wos on soot.mcs.anl.gov, Wed May  3 13:25:19 2000
The command was "otter".  The process ID is 7342.

-----> 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 []
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)))).

In logic, the properties of the axiom set are often of concern: the number of axioms, the length of the longest axiom, and the number of distinct variables in the most complex axiom. As for properties of the proof, in addition to avoiding the use of double negation, the longest formula in the proof contains 30 symbols (excluding the predicate, commas, and parentheses). The single axiom relies on six distinct variables, and three of the deduced steps also rely on six distinct variables, including the first (application of condensed detachment). No less complex proof can be found, as seen by the fact that the first use of condensed detachment requires six distinct variables and by the fact that but a single hypothesis is present. No step requires more than six distinct variables.

IOpen question: Does there exist a proof of lower level than 43 for the Lukasiewicz 23-letter formula?

Open question: Does there exist a proof of less formula complexity (than 30) for the Lukasiewicz 23-letter formula?

One cannot resist wondering about the nature of Lukasiewicz's proof for his single axiom, its properties and its goal in terms of some known axiom system. Perhaps that of Hilbert was the target. We have in hand two length-23 proofs that begin with the three-axiom system of Lukasiewicz and conclude with a deduction of the six-axiom system of Hilbert (one member of which is dependent on the remaining five). (Whether a shorter proof exists is unknown to us.) Therefore, one could simply juxtapose one of the length-23 proofs and the just-given 56-step proof, remove duplicate steps (of which there are three), and have a proof that deduces the Hilbert system from the Lukasiewicz 23-letter single axiom, a proof of length 76. Far more satisfying—especially since the experiment that yielded the 200-step proof of the three-axiom system of Lukasiewicz also found a 194-step proof of Hilbert and found it before finding the 200-step proof—was the prospect of finding a proof whose length is in the neighborhood of 56 (applications of condensed detachment). Indeed, we decided to attack that problem, but, rather than starting with the 194-step proof and applying various proof-shortening techniques of ours, we focused on the 56-step proof (featured earlier) and on a 60-step proof that deduces an axiom system of Church from the 23-letter single axiom. Eventually, OTTER completed the following 59-step proof.

A Short Proof Deducing the Hilbert Axiom System

```
----- Otter 3.0.6, April 2000 -----
The process was started by wos on lemma.mcs.anl.gov, Wed May  3 17:28:13 2000
The command was "otter".  The process ID is 8840.

-----> EMPTY CLAUSE at   1.47 sec ----> 126 [hyper,3,47,120,119,54,93,35]
$ANS(step_allHilbert_18_21_22_3_54_30).

Length of proof is 59.  Level of proof is 44.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] -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).
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))))).
17 [hyper,1,8,8] P(i(x,i(i(i(i(y,z),i(u,z)),i(z,v)),i(w,i(z,v))))).
18 [hyper,1,17,17] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
20 [hyper,1,8,18] 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,1,18,8] 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))).
27 [hyper,1,8,25] P(i(x,i(i(y,i(y,z)),i(u,i(y,z))))).
30 [hyper,1,27,27] P(i(i(x,i(x,y)),i(z,i(x,y)))).
33 [hyper,1,30,30] P(i(x,i(i(y,i(y,z)),i(y,z)))).
35 [hyper,1,33,33] P(i(i(x,i(x,y)),i(x,y))).
39 [hyper,1,35,8] P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(i(z,x),i(u,x)))).
40 [hyper,1,39,39] P(i(i(i(x,y),i(y,z)),i(u,i(y,z)))).
41 [hyper,1,39,35] P(i(i(x,i(i(n(x),n(y)),z)),i(y,i(i(n(x),n(y)),z)))).
42 [hyper,1,39,40] P(i(i(i(x,y),i(z,x)),i(u,i(z,x)))).
43 [hyper,1,35,40] P(i(i(i(x,y),i(y,z)),i(y,z))).
45 [hyper,1,35,42] P(i(i(i(x,y),i(z,x)),i(z,x))).
46 [hyper,1,43,8] P(i(i(i(i(n(x),n(y)),z),x),i(i(x,u),i(y,u)))).
47 [hyper,1,45,43] P(i(x,i(y,x))).
```

48 [hyper,1,45,23] P(i(i(i(i(x,y),z),y),i(x,y))).
49 [hyper,1,46,43] P(i(i(i(n(x),y),z),i(x,z))).
50 [hyper,1,48,46] P(i(i(n(x),n(y)),i(i(x,z),i(y,z)))).
51 [hyper,1,48,42] P(i(x,i(y,i(z,x)))).
54 [hyper,1,49,35] P(i(x,i(n(x),y))).
55 [hyper,1,49,51] P(i(x,i(y,i(z,i(n(x),u))))).
59 [hyper,1,43,55] P(i(x,i(y,i(n(i(z,x)),u)))).
62 [hyper,1,41,59] P(i(x,i(i(n(y),n(x)),i(n(i(z,y)),u)))).
64 [hyper,1,62,41] 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))).
65 [hyper,1,45,64] P(i(n(i(x,y)),n(y))).
66 [hyper,1,50,65] P(i(i(i(x,y),z),i(y,z))).
70 [hyper,1,66,8] P(i(i(i(i(n(x),n(y)),z),x),i(u,i(i(x,v),i(y,v))))).
74 [hyper,1,48,70] P(i(i(n(x),n(y)),i(z,i(i(x,u),i(y,u))))).
75 [hyper,1,39,70] 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)))).
76 [hyper,1,39,74] P(i(i(i(i(x,y),i(z,y)),n(x)),i(u,n(x)))).
78 [hyper,1,75,30] P(i(x,i(i(n(y),n(i(y,z))),i(i(y,z),z)))).
79 [hyper,1,39,76] P(i(i(n(x),i(i(x,y),i(z,y))),i(u,i(i(x,y),i(z,y))))).
80 [hyper,1,78,78] P(i(i(n(x),n(i(x,y))),i(i(x,y),y))).
81 [hyper,1,35,79] P(i(i(n(x),i(i(x,y),i(z,y))),i(i(x,y),i(z,y)))).
82 [hyper,1,49,80] P(i(x,i(i(x,y),y))).
85 [hyper,1,23,82] P(i(i(i(x,i(i(x,y),y)),z),i(u,z))).
86 [hyper,1,75,85] P(i(x,i(i(n(y),n(z)),i(z,i(i(y,u),u))))).
90 [hyper,1,86,86] P(i(i(n(x),n(y)),i(y,i(i(x,z),z)))).
92 [hyper,1,49,90] P(i(x,i(y,i(i(x,z),z)))).
93 [hyper,1,81,92] P(i(i(x,y),i(i(n(x),y),y))).
94 [hyper,1,66,92] P(i(x,i(y,i(i(i(z,x),u),u)))).
96 [hyper,1,81,94] P(i(i(x,y),i(i(i(z,n(x)),y),y))).
97 [hyper,1,85,96] P(i(x,i(i(i(y,n(z)),i(i(z,u),u)),i(i(z,u),u)))).
98 [hyper,1,39,97] P(i(i(i(i(x,y),y),z),i(x,z))).
99 [hyper,1,98,96] P(i(x,i(i(i(y,n(i(x,z))),z),z))).
100 [hyper,1,66,99] P(i(x,i(i(i(y,n(i(i(z,x),u))),u),u))).
104 [hyper,1,42,100] P(i(x,i(i(i(y,n(i(i(z,i(u,v)),u))),u),u))).
107 [hyper,1,39,104] P(i(i(x,y),i(i(i(z,i(x,u)),x),y))).
109 [hyper,1,107,99] P(i(i(i(x,i(y,z)),y),i(i(i(u,n(i(y,v))),v),v))).
111 [hyper,1,39,109] P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))).
113 [hyper,1,111,23] P(i(i(x,y),i(i(y,z),i(x,z)))).
115 [hyper,1,113,113] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
119 [hyper,1,98,115] P(i(i(x,y),i(i(z,x),i(z,y)))).
120 [hyper,1,115,98] P(i(i(x,i(y,z)),i(y,i(x,z)))).

The 59-step proof of Hilbert from the Lukasiewicz 23-letter formula produced for us delight. After all, six members must be deduced in contrast to the three for the Lukasiewicz three-axiom system. Three of its steps rely of six distinct variables, and a thirty-symbol formula is its longest. Most piquant of all, 54 steps of the cited 56-step proof are among the 59 steps of the Hilbert proof. If this observation is considered with the following item, one (such as an expert logician) might discern the basic cause. Specifically, we found a 21-step proof that deduces the Church axiom system from the three-axiom system of Lukasiewicz as well as finding a 23-step proof that deduces the Hilbert axiom system from the same Lukasiewicz system. Of the 21 deduced steps, 17 are among the 23 steps of the Hilbert proof. A quick guess at what might be concluded says that many, many basic lemmas play a key role in such studies.

A Medley of Methodologies

Here we revisit some of the techniques discussed in this chapter and preview some of the techniques presented in the next chapter—methodologies that have proved useful in the successes reported in this volume. OTTER provides the basis for a variety of these methodologies that markedly increase the likelihood of finding missing and elegant proofs. These methodologies rely on diverse strategies (see the sidebar ''Strategy—Key to the Discovery of Proofs'' near the end of Chapter 1). A natural question immediately arises: What is the difference between strategy and methodology? Superficially, the main difference is in *scope*. Typically, a strategy—especially a restriction strategy—reduces the search space of conclusions by more than 50 percent, while a methodology has far less effect.

• *Cramming*: a naive methodology for abridging a proof by compressing into a new proof an existing partial proof together with a few needed, additional steps.

With this technique, OTTER found a 32-step abridgment of the Meredith-and-Prior abridgment of the Lukasiewicz proof for his shortest single axiom for the implicational fragment of two-valued sentential calculus.

• *Symmetric Difference*: a counterintuitive methodology that uses resonators that correspond to the symmetric difference of two proofs with reasonably similar length but sharing roughly half of their deduced steps.

This technique played a key role in OTTER's discovery of a 38-step proof of Meredith's single axiom for the <C,I> calculus.

• *Lemma Adjunction*: an iterative methodology in which one selects certain lemmas (each representable by a clause free of logical **or**), places the negation of each in the passive list, uses the positive form of each as a resonator, and assigns to each resonator a value to reflect its conjectured importance. Chosen lemmas proved in one run are explicitly adjoined as hypotheses in later runs. With this technique OTTER found a proof for the Lukasiewicz 23-letter single axiom.

• *Breadth First*: a methodology in which all immediate children of the statements that present the assignment are deduced before any other conclusion is drawn.

Though often impractical—indeed, leading to an exponential growth in the size of the conclusions to be examined—a breadth-first search proved useful in completing the cited 32-step abridgment and proved useful in obtaining a 10-step proof of the single axiom *XGK* for equivalential calculus.

• *Formula Blocking*: the process of restricting the program's focus by preventing it from accepting one or more formulas or equations.

With this technique, OTTER found a double-negation-free 32-step proof of the dependence of one of Lukasiewicz's five axioms for infinite-valued sentential calculus.

• *Lemma Avoidance*: the process of dispensing with the use of a thought-to-be-indispensable lemma.

With this technique OTTER found a circle of pure proofs for the Moufang identity problem.

Dialogue 2

Chapter 2 (as was the case with Chapter 1) produced questions, answers, and another dialogue among our six protagonists: Ardent Proponent (AP), Strongly Opposed (SO), Curious Interrogator (CI), Interested Mathematician (IM), Practicing Logician (PL), and Reasonable Skeptic (RS). The dialogue began with a heated barrage by SO, in some ways more annoyed than when the six met for the first time (in Dialogue 1).

SO: I am not particularly impressed. Granted they found a missing proof, one that Lukasiewicz chose not to include. However, I am certain the approach they took—especially from the offerings of Chapter 2—bears little resemblance to that taken by Lukasiewicz. Even more, I doubt that any logician has ever taken their approach. If we look into the matter, we shall find that they permitted the program OTTER to deduce millions and millions of conclusions, most of which were useless. And what makes any of you five think that the success reported in this chapter is more than isolated?

RS: Well, SO, you have a friend in court, at least for now. Clearly no logician or mathematician would draw millions of conclusion in an attempt to find a proof. And I doubt if logicians avoid the use of double negation, terms of the form $n(n(t))$. But, since PL is a logician, let us pose that question.

PL: I assume you refer to avoiding double negation. A search of the literature of the masters from Hilbert to Ulrich strongly suggests that avoiding the use of double negation was not part of their approach. In fact, as the authors of this volume note, such avoidance might indeed be counterintuitive. As the authors note, the later chapters and the CD-ROM will present numerous proofs in which negation is present but in which double negation is avoided. One naturally wonders, for propositional calculus, whether an axiom system exists such that a double-negation-free proof can *always* be found when the axioms and the conclusion are free of double negation. As for drawing conclusions in abundance, Lukasiewicz did spend three years of research that culminated in his discovery of his single axiom. Therefore, perhaps he drew an enormous number of conclusions, although millions seems out of the question. What about the OTTER study, AP, that led to the completion of the cited 200-step proof?

AP: I understand that 40,000,000 conclusions were drawn in that study. Since just over 3 CPU-hours were required to complete the success, OTTER must indeed be a very fast program and, more important, be very robust.

SO: Nobody cares any longer about how short a proof is. So why did the authors devote any time and effort to such an activity?

PL: Well, even decades ago, some of the great minds did care—for example, Meredith, Prior, and Thomas. As for who cares here in the year 2001, there may indeed be few—Ulrich certainly cares—but the methodologies developed for that purpose might come in handy in some other context.

CI: Even so, the methodologies developed for that purpose might come in handy in some other context. Perhaps chip designers and program synthesizers could put such methodologies to good use. After all, if you consider a proof that constructs, say, a circuit with n components, and the length of the proof is k with k greater than n, then a proof substantially shorter than k that constructs a circuit satisfying the same specifications might indeed be of interest to a designer of circuits. And as for millions and millions of drawn conclusions, as long as they are sound, following from the assumptions, no harm has been done. I mean, a program doesn't get tired, which is one of the benefits of using OTTER and its like. Besides, from what the authors say, strategies have been formulaed and are being used to keep the corresponding CPU time acceptable. Do any of you five believe the bottom has been reached, that no additional strategic breakthroughs will occur?

SO: It won't make any difference. Most of mathematics will still remain out of reach of automation. Don't you agree, IM?

IM: Actually, if you extrapolate from the mid-1960s, or even from the late 1980s, the future may be unexpectedly exciting, might surprise you, SO. I like the idea of guiding the program with resonators, with formulas or, even better, with equations that the researcher finds appealing. A skilled researcher might, with the aid of OTTER, prove some deep theorems.

CI: What I like is the notion that the program can detect inconsistency within an axiom set supplied by the user, and then detail the inconsistency. From an inconsistent set, one can prove almost anything. In addition to finding an erroneous assumption, can the program be used to detect a missing hypothesis?

RS: I'll take that one, AP. I assume needed but missing hypotheses can be detected by the user by looking for obvious conclusions that should be present but are missing in the output file. Besides, though the admission is given grudgingly, one can draw comfort in the accuracy of OTTER's proofs and in being able to examine in detail the history of each conclusion that is drawn. But don't conclude that I am sold on the automation of reasoning. I am reserving judgment, until more chapters appear.

CI: I, for one, would prefer shorter proofs; I also find convenient fully annotated proofs, and, above all, greatly appreciate the assurance that the proof was free of flaws. I like the fact that the proofs can be independently checked with IVY, if one uses build_proof_object.

AP: I also, to be honest, like the fact that I might make a reasonably significant contribution to logic or to mathematics even though I might know relatively little about the subject. The authors have found missing proofs of significant theorems, and they clearly are not masters. I could, for example, take a few proofs from an expert and use the proof steps in what the authors of this book call lemma adjunction. In a way, the program and I would learn together, neither of us knowing at the start which lemmas could be proved from the given axioms. By taking the steps of an entire proof and using them as resonators, one could instruct or direct the program along a line of reasoning that had proved profitable in the past, rather than directing the program to emphasize the role of isolated facts.

PL: Good observation; in fact, using Lukasiewicz's 68 theses as resonators clearly corresponds to relying on a path of reasoning that merited publication. Whether Lukasiewicz relied on them to obtain a proof for his single axiom remains a mystery.

SO: I am quite suspicious. How many of the 68 formulas are among the 200 in the proof yielded by the given methodology? And, if those 68 are viewed as resonators, how many of the 200 steps match at least one of the 68 theses, in other words treating all variables in both sets of formulas as indistinguishable?

AP: If you examine the 200-step proof found on the CD-ROM, you'll find that only 8 of the 68 are among the 200 steps, which might weakly suggest that Lukasiewicz pursued some other line of thought. Only 22 of the 200 match at the resonator level (ignoring specific variables) at least one of the 68.

CI: You know, until now, I thought of propositional calculus as offering nothing deep to study. I thought it was all about truth tables and little else.

PL: Far from it! As this chapter suggests, this area of logic presents some deep questions to attempt to answer. For example, does there exist a single axiom for the calculus with strictly fewer than 21 symbols, the length of the Meredith single axiom? Then there are questions focusing on double negation and questions focusing on the number of distinct variables needed to complete a proof.

IM: Wait, before you go on, PL, the study of single axioms of minimal length can be most intriguing. For example, OTTER played a key role in finding a shortest single axiom in the Sheffer stroke for Boolean algebra, and it was heavily used to find an impressively short single axiom for that area where the Sheffer stroke was replaced by the functions **or** and **not**. In particular, there exists an equation consisting of 15 symbols in the Sheffer stroke that is a single axiom for Boolean algebra, and no shorter axiom exists. Regarding **or** and **not**, OTTER found a length-22 single axiom. Whether a shorter axiom exists remains open at this time.

AP: I understand that this volume will contain at least one chapter on equality and focus on applying OTTER-oriented methodology that yields short proofs for these axioms.

CI: Tell us more, PL, about the deep questions focusing on double negation and on the number of distinct variables.

PL: Let's focus on the latter first, for the former leads to an interesting story, so I am told. Regarding the number of distinct variables, one might make the following conjecture for propositional calculus; its consideration provides a taste of the depth offered by this area of logic. Let A be an axiom system, and let j be the maximum number of distinct variables in at least one of its members. Let F be a set of formulas provable from A by condensed detachment alone, and let k be the maximum number of distinct variables in at least one member of F. Conjecture: One can always find a proof relying on condensed detachment alone that deduces all of F from A such that none of its deduced steps requires the use of strictly more distinct variables than the maximum of j and k. By the way, the conjecture holds if condensed detachment is replaced by two inference rules, detachment (modus ponens) and substitution (instantiation); that pair of inference rules behaves quite differently than does condensed detachment. However, thanks to Ulrich, we know that the conjecture can be refuted with a two-axiom system in part relying on Meredith's single axiom. He shows that one can provide an example requiring as many variables as one wishes.

CI: What about a deep question focusing on double negation, PL?

PL: As posed by Ulrich, does there exist an axiom system *A* for two-valued sentential (or propositional) calculus such that, first, all members of *A* are free of double negation and, second, whenever a set *f* of formulas provable from *A* contains no formula in which double negation occurs, one can always find a double-negation-free proof of *F* from *A* relying solely on condensed detachment? Ulrich has proved that axiom systems *A* exist that are free of double negation, but for which double-negation free *F* exist such that *F* is provable from *A* and such that all proofs of *F* from *A* relying solely on condensed detachment require the use of double negation, or far worse.

SO: I doubt that OTTER could provide much assistance in trying to answer Ulrich's open question.

PL: Well, SO, that's where the story I mentioned begins. Michael Beeson learned of the question and began working on it. I understand that he is close to proving that the Lukasiewicz three-axiom system, all of whose members obviously are free of double negation, has the desired property. Specifically, if you take a set *F* of formulas such that each is free of double negation and such that each is provable from the Lukasiewicz system, you can always find a proof relying solely on condensed detachment in which no step contains a term of the form $n(n(t))$. I have also heard—and this might interest you, SO, especially—that OTTER is playing a key role. Perhaps AP can shed some light on the situation and complete the story.

AP: OTTER is certainly involved in Beeson's attempt. In fact, he sent Wos twenty-two lemmas with the request that OTTER find double-negation-free proofs of each. Using one of the standard input files that relies on the resonance strategy and that uses demodulation to block retention of any deduced conclusion in which double negation is present, Wos submitted the lemmas to OTTER for study. Rather quickly, OTTER proved seventeen of the lemmas but got no further. Then, in a second run whose objective was to prove the remaining five lemmas, Wos used lemma adjunction, adjoining to the set of support all of the proof steps of the corresponding seventeen proofs, of course removing duplicate steps. OTTER worked rather hard: In just over 6 CPU-hours on a slow machine (Pentium II, 400 MHz), the program proved four of the five. But the fifth remained out of reach. So a breadth-first search, level saturation with the command set(sos_queue), was brought into play. In 10 CPU-minutes and with the retention of clause (89548), the fifth and last lemma was proved.

CI: Was the proof long, and did it involve complex formulas that were never chosen as the focus to initiate the use of condensed detachment?

AP: Complex formulas that are deduced and retained can, because of their complexity (measured in symbol count), wait forever to be considered by OTTER because of its algorithm for choosing which clause to be the next on which to focus. As it turned out—brace yourselves—the proof consists of two lines, the second of course being the sought-after lemma. Further, the first line of the proof has weight 14, consists of but 14 symbols.

RS: Then why wasn't it considered in one of the earlier runs? After all, 14 is not a very large number; I mean, the clause is not particularly complex.

AP: When the earlier runs were examined, the key first clause of the two-line proof was retained at about clause (14000) or so. It was not chosen, even though the program was choosing clauses numbered 181,000 and greater, partly because of the resonators that caused various retained conclusions to be assigned a weight (complexity) strictly less than 14 and partly because of the presence of so many clauses with complexity less than 14 when measured in symbol count.

RS: And so we have uncovered a weakness of OTTER, namely, its inability to focus on a rather innocent-looking clause. Even with the intriguing arsenal of strategies offered by OTTER, they proved insufficient.

IM: Many of my students suffer from the same weakness; so do many of my colleagues, occasionally. Nor am I, sad to say, immune from overlooking a promising path of reasoning.

CI: That's one of the obstacles that faces automated reasoning, and it's one of the motivations for formulating additional strategies. But wait; why bother with lemma adjunction, and instead why not simply include the proof steps as resonators? When the program proves a lemma matching one of the resonators, it will focus on it rather quickly because the clause-equivalent of the lemma will be assigned a small weight.

AP: The situation is somewhat like the one just discussed in the context of the first step of the two-step proof. Indeed, if a lemma is proved that matches a resonator but is thus given a small weight (high priority for consideration), and if the corresponding clause is numbered, say, 14,000, the lemma may wait a long, long time—perhaps essentially forever—because of the presence of so many earlier clauses that are given preference. On the other hand, when the lemma is included in the input in the (initial) set of support and the program is told to focus on all the clauses in the input set of support first, the difficulty just illustrated is removed. So you see why lemma adjunction can be most profitable and often not replaceable by the appropriate resonators.

SO: So, AP, I see that the use of resonators can be most dangerous. If a resonator is matched by many, many clauses, the clause that proves key to unlocking the proof may never be considered. So much for automated reasoning!

IM: You're not totally convincing, SO. Various mathematicians and various logicians with no interest in research in automated reasoning successfully use the resonance strategy. On a more playful note, a person could apply the methodology given in this chapter, stopping when the desired formulas or equations have been proved. Then, by extracting all of the proofs of all of the relevant lemmas, one could publish a possibly lengthy and, to those less familiar with research that make judgments based on length, impressive paper. In other words, one could simply omit the use of what the authors of this book call proof contraction. After all, we all know how much praise is sometimes heaped upon a researcher who proves an interesting theorem, where the proof rests on the use of, say, twenty or more lemmas also proved. A short paper offering a short proof can mislead some about the difficulty of having found the proof. Also well known is the fatigue that can occur after working for months or years (as Lukasiewicz did with his single axiom) on some interesting result, a genuine exhaustion that could make virtually impossible the attempt to contract by hand a lengthy proof dependent on many lemmas. Of course, and I am certain that we six agree, one is well advised to offer (if energy permits) the shorter paper with the more elegant proof. In that regard, I find the discussion of proof contraction implying in no way disrespect or a lack of appreciation for the marvelous contributions of a skilled researcher, and I look forward to the chapter or chapters that offer means for a program to find shorter proofs.

CI: Would somebody explain again the so-called before and after of proof contraction, summarize the differences between contraction and proof shortening, and compare a contracted proof with the type found in the literature?

AP: I'll take this one. **AP**: OTTER's proofs (as seen in this chapter) take a standard form. They begin with statements, clauses, from the input, followed by a set of deduced conclusions without interruption, where each deduced conclusion is preceded by its parents (by number) and the inference rule used to obtain it. The input statements correspond to the hypotheses (axioms) of the theorem being proved and, possibly, clauses for the inference rules in use (such as condensed detachment). If one or more axioms is not present in a proof, those that are present suffice. Now for a proof to be considered for contraction, before the deduced conclusions are given, one finds clauses that were not among the original items that presented the theorem. Those so-called extra clauses correspond to lemmas that were proved in an earlier run, showing that lemma adjunction was used along the way. Proof contraction is designed to produce a proof in which no extra clauses are present. Of course, some or all of the lemmas that were present in the proof to be contracted may be present among the deduced conclusions after contraction.

IM: But—and here is a charming feature of proof contraction that appeals to me as a mathematician—one or more of the lemmas proved along the way and listed in the uncontracted proof may be totally absent after contraction. If that happens for a given lemma, one learns that, although the lemma originally was needed and used, it is no longer needed. If one decided to skip contraction and be satisfied with the proof that was discovered (including the extra lemmas), then one must include as part of the final result proofs of the added lemmas. Further, if in the proofs of any of the added lemmas OTTER listed other lemmas proved yet earlier, then of course one would be required to include their proofs as well. One would (in effect) be traversing in reverse the path that is taken during research that produces the typical published paper. A contracted proof, because it is not preceded by one or more proofs of needed lemmas, is atypical of that found in the literature. Any needed lemmas, as well as their proofs, are simply embedded in a contracted proof, which is fine with me. Conversely, one could take a standard OTTER proof and turn it into a proof reminiscent of the literature by identifying the key lemmas and breaking out their proofs separately, ignoring the fact that some proof steps might appear in one or more of the lemmas.

AP: As for proof length, OTTER merely gives the length of the so-called final proof, not counting the extra lemmas that may be present. To compute the actual length, one would be forced to follow a path similar to that just described by IM, examining all of the needed lemmas proved earlier and adjusting the sum to account for duplicate steps. The length of the contracted proof will, in most cases, be greater than the list length of the uncontracted proof (not taking into account the lengths of the proofs of the extra lemmas). But it will most likely be smaller than the ''actual'' length of the uncontracted proof when one considers the extra lemmas that were used, in fact, possibly much smaller. Proof shortening differs from proof contraction in various ways. First, the proof being shortened contains no extra lemmas, clauses that are listed before the deduced conclusions and that are not from the original theorem under study. Second of all, a concrete effort is made to find a proof of smaller length, in contrast to contraction that just happens often to find one of shorter length. For example, with contraction, the parameter values are not changed nor the options used in the discovery of the uncontracted proof. With proof shortening, different combinations of options are tried and different assigned values are employed, such as experimenting with a new value for the number of distinct variables allowed in a deduced-and-retained conclusion.

CI: On the subject of the number of distinct variables required to complete a proof, I like the fact that the 56-step proof relies on formulas with six distinct variables and no more. After all, a greater number could be required for such a deep theorem, and fewer than six is impossible, as one can see from the fact that the first condensed detachment step requires six, and only one hypothesis is available, the Lukasiewicz 23-letter formula.

PL: An interesting observation, CI. Note that the Meredith single axiom relies on five distinct variables, and I happen to know that the first step of any proof with it as sole hypothesis relies on five variables. Therefore, any proof of any type with this axiom as sole hypothesis requires at least five distinct variables. Now, Meredith's 41-step proof contains two formulas, each relying on seven variables, which you can verify with the CD-ROM. Some of you might be intrigued to learn that logicians are interested in the number of axioms in an axiomatization, the maximum number of distinct variables in the members, and the length of the longest. They also are interested in the length of proofs; for example, Meredith and Prior published what they called an ''abridgment'' of a proof of Lukasiewicz, a slightly shorter proof than he had offered. What I'm leading up to is a guess about two types of missing proof that the authors of this book will present, perhaps shorter proofs and perhaps proofs relying on fewer distinct variables.

AP: OTTER offers direct assistance for each type. To search for shorter proofs, one can use set(ancestor_subsumption), a command that compares derivations of the same conclusion, preferring the shorter. Regarding distinct variables, one can use assign(max_distinct_vars,k), a command that automatically purges any deduced conclusion with greater than *k* distinct variables in it. Where the maximum number of distinct variables in the deduced steps of a proof is the measure, I know that OTTER found an improvement on Meredith's 41-step proof, specifically, a proof requiring six distinct variables. In particular, by building on the research of a colleague Deepak Kapur, a 54-step, six-variable proof was discovered; it is included on the CD-ROM in the book *A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning* (authored by Larry Wos and Gail Pieper, and published by World Scientific). So I'm certain, PL, that the two types you suggest will play an important role in this volume. I wonder whether a five-variable proof exists for the Meredith single axiom.

IM: Off the topic, I think the notion of axiomatic proof is becoming clearer. The two proofs given in this chapter rely on a given inference rule, condensed detachment, and all steps are formulas within the area under study. In contrast, from what I have heard, proofs in which *equality* is the only relationship will rely on paramodulation alone, although canonicalization and simplification may be employed through the use of demodulation, and all deduced steps will be equations from the theory under consideration. For example, no mention of the integers will occur in a proof, unless integers are directly in focus. Also, pleasing to logicians and not so typical of mathematics, each proof will explicitly cite the inference rules used—hyperresolution when condensed detachment is the concern—and all parents will be explicitly listed along with an explicit listing of the demodulators that are used. As the authors note, induction will never be used in the proofs they give in this volume; OTTER simply does not offer it.

RS: Which is why SO maintains that most of mathematics will remain out of reach of their research. Perhaps, in the not-too-distant future, some clever young researcher will connect a program offering induction with OTTER, and then we shall see what happens. Nevertheless, since they intend to give many, many proofs, being able to dispense with induction makes their work rather interesting and, perhaps, rather impressive; after all, the literature of proofs is speckled with the use of induction, of higher-order logic, Zorn's Lemma, and the like. For now, let us adjourn, meeting for a third dialogue after we all read Chapter 3. Since the authors referred in Chapter 1 to this book as (in one sense) an encyclopedia of missing proofs, the next chapter will surely offer such and, perhaps, the methodology used to discover them.