papers/notebooks/bcsk
created 05-08-2005
revised last 06-23-2009

A Prelude to Unveiling Some of the Mysteries of Proof Discovery: A Study of the BCSK Logic*

Larry Wos

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

1. Prelude, Perspective, Preamble, and Possibilities

The basis of this notebook is a detailed study, prompted by Matthew Spinks, of an area of logic that was totally new to me, an area previously studied by Spinks and Robert Veroff. I could not have predicted what was to come from the study, especially the proving of various new theorems concerning unexpected relationships among the axioms of concern. These new theorems, from what I know, produced genuine surprise to some already in the field of logic in focus, namely, the *BCSK* logic, whose axioms I given in Section 2. I invite you to experience the pleasure, and even astonishment, I experienced and, perhaps, accept some of the diverse challenges that are offered. Further, this notebook will, for some or many, play the role of a syllabus or precis regarding research and regarding the use of a powerful automated reasoning program, namely, W. McCune's OTTER. The copious detail given here may even be edifying to various researchers who have been using OTTER for some time.

Why are certain people good at chess or at poker? Have you ever wondered what enables a person to win so often at word games such as scrabble? How does a mathematician or a logician find a proof? (When I was a student in the Mathematics Department at the University of Chicago, I asked Paul Halmos questions intended to learn his secrets for proof finding; I did not succeed. I believe now he could not answer such questions, for they are in the most obvious sense unanswerable.) Can an expert in some field of mathematics or logic learn of new theorems whose discovery was made possible by a computer program that reasons logically? And, so relevant to this notebook or essay (as well as other notebooks found on my website, automatedreasoning.net), I pose the following question: Which approaches taken by an automated reasoning program are most effective at traversing the potentially huge space of deducible conclusions in search of a first proof, or a new theorem, or a better and more elegant proof than currently in hand?

By focusing most heavily on the third item, finding more elegant proofs, I plan to shed some light and, indirectly, begin the journey that may culminate in a far fuller understanding of the nature of proof and of the space of conclusions that can be explored. In this notebook and (if all goes as planned) a number of other notebooks, I shall tell a story that details various experiments that led to new results, some of which concern unexpected relationships among axioms and some of which concern proofs more elegant than previously known. As for the unexpected relationships, in the main focusing on axiom dependencies, (from what I know) they were unknown to some of the experts. If such is the case, which I have been led to believe, then, contrary to the position taken by some in the *New York Times* more than fifteen years ago, you can learn—and learn much—from computer proofs.

---

*This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

Of the various ways a new proof can be more elegant (or better) than that in hand, perhaps the most obvious concerns proof length. By proof length, I mean the number of deduced steps in the proof; in other words, the axioms and lemmas and clauses for inference rule(s) in the input are not counted. Further, deduced steps that are implicit are not allowed in the count, steps such as those arising from the use of demodulation; indeed, obviously their presence would not contribute to the given length, making that quantity inaccurate. (In a proof whose length is given, steps may result, for example, by applying hyperresolution or paramodulation. When the inference rule condensed detachment, whose clause is given shortly, is in use, proof length measures the number of applications of condensed detachment; for this essay, that number measures the use of hyperresolution. When comparisons are made regarding proof length, to be fair, the playing field must be reasonably even. Among the implicit rules of play to follow, the same axioms, inference rules, and the like must be present for the comparison. If, for example, you omit a clause that corresponds to an inference rule, then the proof obtained might indeed pay a penalty in length. For a second example, if additional information is supplied in the form of axioms or lemmas, then the resulting proof might well be expected to be shorter than without their presence. In short, for a comparison of proof lengths, a cursory glance at that part of the proof that excludes the deduced clauses should reveal identical sets of formulas or equations, although the order might be different.)

For condensed detachment, when implication is the key operator, the following clause is included, where "-" denotes logical **not**, " | " denotes logical **or**, and the predicate *P* denotes "provability". All variables, such as *X*, *y*, and *z* are implicitly universally quantified, meaning "for all".

```
-P(i(x,y)) | -P(x) | P(y).  % condensed detachment
```

As you will soon see, the syntax changes when, for example, equivalence is in focus.

This essay will feature means for refining proofs, mainly, but not exclusively, in the context of proof length. But, how hard is it to take a proof and find a shorter one? Of course, seeking shorter proofs implicitly means that a given axiom set is in focus; for example, with a richer axiom set, in general, a person or a program should be able to find shorter proofs than with a less rich set. And can the process of proof refinement be automated? Further, of what use is proof shortening, other than aesthetic? I shall take these questions in reverse order and, while doing so, offer topics for research or for doctoral theses or for the joy of the pursuit.

Imagine that you have a set of methodologies that very often, or quite often, take a proof or set of proofs as input and return a set of proofs of the same theorems, but proofs each of which is shorter, perhaps much shorter, than originally in hand. With these methodologies, you could take a standard text, say in logic, and perhaps markedly trim its size (measured in pages), producing a book on the same topic that offers the same theorems but with economy. For example, the original treatment may include lemmas (with their proofs) that are apparently needed, lemmas present mainly or only because they are used to prove interesting theorems. Your success with proof shortening would, in many cases, obviate the need for some of these lemmas. If you succeed in the cited endeavor, then a shorter, more compact, and easier-to-read book might result. What an achievement!

As for automation, I first note that no practical algorithm (from what I know) exists for finding shorter proofs or, for that matter, for finding the shortest proof for a given theorem. (I shall return to this observation in more detail later.) Nevertheless, I am certain that a skilled programmer could study what follows and produce a program that automates much of what is offered in this (and related) notebooks. Such a program would take as input a theorem with a given proof and, if successful, return a proof of strictly smaller length. Most likely, iteration would be at the heart of the program. (As William McCune, author of OTTER, observes, more important than any specific success—even if the result is deep—is the approach taken, the details of how the success was obtained; his observation explains in part why this notebook is being written. OTTER is the automated reasoning program featured here and elsewhere on my website. You need not be an expert in its use to derive great satisfaction and make significant discoveries.) Indeed, one proof after another would be found and, among them, in each run, if all goes well, you might extract a sequence of proofs of descending lengths. The program would with almost certainty be required to branch over and over, sometimes even backtracking. In addition to its use in the preceding endeavor (about book reduction), the program might be valuable for finding better circuits or better computer code.

If you wonder about how hard it is to shorten a given proof, the following might prove instructive and challenging. The area is equivalential calculus, but you need know nothing about the field to accept the challenge. I shall give you an input file that proves the theorem in focus; you are asked to take the proof that results from its use, which I shall also include here, and find a much shorter proof than that which I include and that which is the one obtained from the use of the input file. You will see that the inference rule in use is hyperresolution, which together with the first clause found in list(usable) has the program apply condensed detachment. A bit of background is in order, although you can skip the next few sentences and simply take the input file and run it with McCune's marvelous automated reasoning program OTTER. Or, instead of reading the next few sentences and running OTTER or just running OTTER with no prior reading, you can attempt to duplicate what I am about to tell you *without* looking at the input file.

For seven decades, researchers did not know whether the following formula, *XCB*, was a single axiom for equivalential calculus.

P(e(x,e(e(e(x,y),e(z,y)),z))). % XCB

One way to show that it is in fact a single axiom is to apply condensed detachment (see the input file, first clause in the usable list) enough times to deduce some other axiom system for equivalential calculus.

-P(e(x,y)) | -P(x) | P(y). % condensed detachment

You will find in list(passive) the negations of various single axioms as well as the negations of reflexivity, symmetry, and transitivity. (Although not obvious, reflexivity is dependent on the 2-basis consisting of symmetry and transitivity. In the context of short proofs, the presence of added axioms, such as dependent ones, likely will enable a person or program to find shorter proofs rather than longer; see Section 5.) Because you may wish to seek your own proof that *XCB* is a single axiom without being influenced by the approach that succeeded, here are some known single axioms, in negated form.

```
-P(e(e(a,b),e(e(c,b),e(a,c)))) | $ANSWER(P1_YQL).
-P(e(e(a,b),e(e(a,c),e(c,b)))) | $ANSWER(P2_YQF).
-P(e(e(a,b),e(e(c,a),e(b,c)))) | $ANSWER(P3_YQJ).
-P(e(e(e(a,b),c),e(b,e(c,a)))) | $ANSWER(P4_UM).
-P(e(a,e(e(b,e(a,c)),e(c,b)))) | $ANSWER(P5_XGF).
-P(e(e(a,e(b,c)),e(c,e(a,b)))) | $ANSWER(P7_WN).
-P(e(e(a,b),e(c,e(e(b,c),a)))) | $ANSWER(P8_YRM).
-P(e(e(a,b),e(c,e(e(c,b),a)))) | $ANSWER(P9_YRO).
-P(e(e(e(a,e(b,c)),c),e(b,a))) | $ANSWER(PYO).
-P(e(e(e(a,e(b,c)),b),e(c,a))) | $ANSWER(PYM).
-P(e(a,e(e(b,e(c,a)),e(c,b)))) | $ANSWER(XGK).
-P(e(a,e(e(b,c),e(e(a,c),b)))) | $ANSWER(XHK).
-P(e(a,e(e(b,c),e(e(c,a),b)))) | $ANSWER(XHN).
```

As suggested—and be warned that this challenge is potentially gigantic—you could try to start with *XCB* and seek a proof that completes with the deduction of another known single axiom or completes with the deduction of both symmetry and transitivity, both of which are given in positive form almost immediately. Not surprising, such a deduction is possible, or I probably would not be telling this story. After applying various methodologies, yes, OTTER completed an appropriate proof, one that deduces both symmetry and transitivity.

```
P(e(x,x)). % reflexivity
P(e(e(x,y),e(y,x))). % symmetry
P(e(e(x,y),e(e(y,z),e(x,z)))). % transitivity
```

(Intuitively, you might guess that an axiom system for equivalential calculus, because of its name, consists of the just-given three formulas. It does, but, as it turns out—and you might enjoy trying for the corresponding proof—reflexivity can be deduced from the other two members of the threesome by using condensed detachment.) The answer, in the affirmative (obtained by OTTER, Ted Ulrich, and me), to that long-standing open question produced excitement for me and for my colleagues: The first proof showing

XCB to be a single axiom completed with the deduction of first symmetry, then transitivity. Iteration was the key, finding one important subproof after another.

Here is the promised input file, followed by the proof of concern.

Input File for the XCB Challenge

```
set(hyper_res).
assign(max_mem,480000).
% set(sos_queue).
assign(max_weight,64).
assign(max_proofs,-1).
assign(pick_given_ratio,2).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).
set(order_history).
clear(print_kept).

weight_list(pick_and_purge).
weight(P(e(e(x,y),e(e(y,z),e(x,z)))),6).
end_of_list.

list(usable).
-P(e(x,y)) | -P(x) | P(y).
-P(e(e(a,b),e(b,a))) | -P(e(e(a,b),e(e(b,c),e(a,c)))) | $ANSWER(all_s_t_indep).
-P(e(a,a)) | -P(e(e(a,b),e(b,a))) | -P(e(e(a,b),e(e(b,c),e(a,c)))) | $ANSWER(all_r_s_t).
end_of_list.

list(sos).
P(e(x,e(e(e(x,y),e(z,y)),z))). % XCB
end_of_list.

% list(demodulators).
% (e(e(x,x),y) = junk).
% (e(y,e(x,x)) = junk).
% (e(x,junk) = junk).
% (e(junk,x) = junk).
% (P(junk) = $T).
% end_of_list.

list(passive).
% Following are negations of the 15 length 7 theorems.
-P(e(e(e(a,b),a),b)) | $ANS(f7a).
-P(e(e(a,e(b,a)),b)) | $ANS(f7b).
-P(e(e(a,b),e(a,b))) | $ANS(f7c).
-P(e(a,e(e(b,a),b))) | $ANS(f7d).
-P(e(a,e(b,e(a,b)))) | $ANS(f7e).
-P(e(e(e(a,b),b),a)) | $ANS(f7f).
-P(e(e(a,e(b,b)),a)) | $ANS(f7g).
-P(e(e(a,b),e(b,a))) | $ANS(f7h).
-P(e(a,e(e(b,b),a))) | $ANS(f7i).
-P(e(a,e(b,e(b,a)))) | $ANS(f7j).
-P(e(e(e(a,a),b),b)) | $ANS(f7k).
-P(e(e(a,e(a,b)),b)) | $ANS(f7l).
```

```
-P(e(e(a,a),e(b,b))) | $ANS(f7m).
-P(e(a,e(e(a,b),b))) | $ANS(f7n).
-P(e(a,e(a,e(b,b)))) | $ANS(f7o).
% Following are axioms for EC and other targets.
% negations of known theorems and axioms
-P(e(e(a,b),e(e(c,b),e(a,c)))) | $ANSWER(P1_YQL).
-P(e(e(a,b),e(e(a,c),e(c,b)))) | $ANSWER(P2_YQF).
-P(e(e(a,b),e(e(c,a),e(b,c)))) | $ANSWER(P3_YQJ).
-P(e(e(e(a,b),c),e(b,e(c,a)))) | $ANSWER(P4_UM).
-P(e(a,e(e(b,e(a,c)),e(c,b)))) | $ANSWER(P5_XGF).
-P(e(e(a,e(b,c)),e(c,e(a,b)))) | $ANSWER(P7_WN).
-P(e(e(a,b),e(c,e(e(b,c),a)))) | $ANSWER(P8_YRM).
-P(e(e(a,b),e(c,e(e(c,b),a)))) | $ANSWER(P9_YRO).
-P(e(e(e(a,e(b,c)),c),e(b,a))) | $ANSWER(PYO).
-P(e(e(e(a,e(b,c)),b),e(c,a))) | $ANSWER(PYM).
-P(e(a,e(e(b,e(c,a)),e(c,b)))) | $ANSWER(XGK).
-P(e(a,e(e(b,c),e(e(a,c),b)))) | $ANSWER(XHK).
-P(e(a,e(e(b,c),e(e(c,a),b)))) | $ANSWER(XHN).
-P(e(a,a)) | $ANSWER(reflex).
-P(e(e(a,b),e(b,a))) | $ANSWER(symm).
-P(e(e(a,b),e(e(b,c),e(a,c)))) | $ANSWER(trans).
-P(e(e(e(c1,c2),c3),e(c1,e(c2,c3)))) | $ANSWER(Wajsberg_1).
-P(e(e(c1,e(c2,c3)),e(c3,e(c2,c1)))) | $ANSWER(Wajsberg_3).
-P(e(e(e(c1,e(c2,c3)),e(e(c3,c4),c4)),e(c1,c2))) | $ANSWER(Wajsberg_4_sing).
-P(e(e(e(e(c1,c2),c3),c4),e(c4,e(c1,e(c2,c3))))) | $ANSWER(Wajsberg_5_sing).
-P(e(e(c1,e(c2,c3)),e(e(c2,e(c4,c3)),e(c4,c1)))) | $ANSWER(Bryman_sing).
-P(e(e(c1,e(c2,c3)),e(e(c2,e(c3,c4)),e(c4,c1)))) | $ANSWER(Luka_1_sing).
-P(e(e(c4,e(c1,e(c2,c3))),e(e(c1,c2),e(c3,c4)))) | $ANSWER(Luka_2_sing).
-P(e(e(c1,e(c2,c3)),e(e(c1,e(c3,c4)),e(c4,c2)))) | $ANSWER(Sobo_1_sing).
-P(e(e(c1,e(c2,c3)),e(e(c1,e(c4,c3)),e(c4,c2)))) | $ANSWER(Sobo_2_sing).
end_of_list.

list(hints).
% Following Proof of Reflexivity from XCB
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v)).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),e(v6,w)),v6)).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v))).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v6))).
P(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w)),x)).
P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),e(v7,v6)),v7),v8),v9),
   e(v8,v9)))),v10),e(v11,v10)),v11)).
P(e(e(x,e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),e(e(e(v6,e(e(e(v6,v7),e(v8,v7)),v8)),v9),
   e(v10,v9)))),v10)),x)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),e(v7,v6))),v7)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u)).
P(e(x,x)).
% Following 13 prove a generalization of a Wajsberg, from temp.xcb.exp2.out1
P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v)),w),e(v6,w)),v6)).
P(e(e(e(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w)),x),v6),e(v7,v6)),v7)).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(w,v)),w)).
P(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x))).
P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),v),e(w,v)),w)).
```

P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),w),e(v,w))).
P(e(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u)),x)),v),w),e(v,w))).
P(e(x,e(e(y,e(z,e(e(e(z,u),e(v,u)),v)),y)),x))).
P(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),e(v,e(e(e(v,w),e(v6,w)),v6)))).
P(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),x)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(e(e(u,v),e(w,v)),w))).
P(e(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),e(v,e(e(e(v,w),e(v6,w)),v6))),v7),e(v8,v7)),v8)).
P(e(e(e(x,y),x),y)).
% Following 71 prove f7b and f7e, queue, from temp.xcb.exp1.out1.
P(e(e(e(e(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(w,v)),w),v6),e(v7,v6)),v7),v8),e(v9,v8)),v9)).
P(e(e(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v6)),v7),e(v8,v7)),v8)).
P(e(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(w,v)),w),v6),e(v7,v6)),v7)).
P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(e(e(u,v),e(w,v)),w))),v6),e(v7,v6)),v7),v8),e(v9,v8)),v9)).
P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),e(v6,w)),v6),e(e(e(v7,e(e(e(v7,v8),
e(v9,v8)),v9)),v10),v11),e(v10,v11)))).
P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v6)),e(e(e(v7,e(e(e(v7,v8),
e(v9,v8)),v9)),v10),v11),e(v10,v11)))).
P(e(e(e(e(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),x),v),e(w,v)),w),v6),e(v7,v6)),v7)).
P(e(e(e(e(e(e(x,e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x),v6),e(v7,v6)),v7)),v8),v9),
e(v8,v9)),v10),e(v11,v10)),v11)).
P(e(e(e(e(e(e(x,e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x),v6),e(v7,v6)),v7)),v8),v9),
e(v8,v9)),v10),v11),e(v10,v11))).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),e(e(e(e(w,v6),w),v6),v7)),v8),e(v7,v8))).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(v,e(e(e(v,w),e(v6,w)),v6))),v7),e(v8,v7)),v8)).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),w),e(v,w)),e(e(e(v6,e(e(e(v6,v7),e(v8,v7)),v8)),v9),v10),
e(v9,v10)))).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v)),w),e(v6,w)),v6)).
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v)),w),v6),e(w,v6))).
P(e(e(e(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),x),v),e(w,v)),w),e(e(e(e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)),v9),v9),v10),v11)),e(v10,v11))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),e(e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v8),v9)),e(v8,v9))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),e(e(e(w,v6),e(v7,v6)),v7))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(v,e(e(e(v,w),e(v6,w)),v6))),v7),e(e(e(v7,v8),e(v9,v8)),v9))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(w,v)),w)).
P(e(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),e(v,e(e(e(v,w),e(v6,w)),v6))),v7),v8),e(v7,v8)),
e(v9,e(e(e(v9,v10),e(v11,v10)),v11)))).
P(e(e(e(e(e(e(x,y),x),y),z),e(u,z)),u)).
P(e(e(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w)),x),v6),e(v7,v6)),v7)).
P(e(e(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),w),e(v,w))),x),v6),e(v7,v6)),v7)).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),v6),v7),
e(v8,v7)),v8)),v9),e(v10,v9)),v10)).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(u,e(e(e(u,v),e(w,v)),w)),v6),v6)),v7),e(v8,v7)),v8)).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(u,e(e(e(u,v),e(w,v)),w)),v6),v6)),v7),v8),e(v7,v8))).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(e(e(u,v),e(w,v)),w))),v6),e(v7,v6)),v7)).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),e(e(e(e(e(e(w,e(e(e(w,v6),
e(v7,v6)),v7)),v8),e(v9,v8)),v9),v10),v11),e(v10,v11)))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),e(e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v8),v9),e(v8,v9)))).
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(e(e(v,w),e(v6,w)),v6))).
P(e(e(e(e(e(x,e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),v)),x),w),e(v6,w)),v6)).
P(e(e(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v8),v8)),v))),x),v9),
e(v10,v9)),v10)).
P(e(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),v),w),e(v,w)),e(v6,e(e(e(v6,v7),e(v8,v7)),v8)))).
P(e(e(e(e(e(x,e(y,e(e(e(e(e(z,e(e(e(z,u),e(v,u)),v)),e(w,e(e(e(w,v6),e(v7,v6)),v7)),y)),v8),

e(v9,v8)),v9))),x),v10),e(v11,v10)),v11)).
P(e(e(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),x),v),e(w,v)),w)).
P(e(e(e(e(x,e(e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x),v6),e(v7,v6)),v7)),v8),e(v9,v8)),v9)).
P(e(e(e(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w)),e(e(e(e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)),v9),v9),v10)),v11),e(v10,v11))).
P(e(e(e(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w)),v6),e(v7,v6)),v7)).
P(e(e(e(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w)),v6),v7),e(v6,v7))).
P(e(e(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),w),e(v,w))),x),e(e(e(e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)),v9),v9),v10),e(v11,v10))),v11)).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(u,e(e(e(u,v),e(w,v)),w)),e(e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)),v9))),v10),e(v9,v10))).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(u,e(e(e(u,v),e(w,v)),w)),e(e(v6,e(e(v7,e(e(e(v7,v8),
e(v9,v8)),v9)),v6)),v10))),v11),e(v10,v11))).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(e(e(e(e(e(v,e(e(e(v,w),e(v6,w)),v6)),v7),v7),v8),v9),e(v8,v9)))).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(e(e(e(e(v,e(e(e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v8),v9),
e(v8,v9))),v),v10),v11),e(v10,v11)))).
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(v,e(e(e(v,w),e(v6,w)),v6)))).
P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v8),v8)),v))),x),
e(e(e(v9,v10),e(v11,v10)),v11)),v9)).
P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v)),x))),v8),e(v9,v8)),v9)).
P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),v),e(w,v)),w)).
P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),v),w),e(v,w))).
P(e(e(e(e(x,y),e(z,y)),z),x)).
P(e(e(e(x,e(e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x),v6),e(v7,v6)),v7)),v8),v8)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),v6),v7),e(v8,v7))),v8)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(e(e(e(e(e(v,e(e(e(v,w),e(v6,w)),v6)),v7),v7),v8),v9),e(v8,v9))))),u)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(v,e(e(e(v,w),e(v6,w)),v6)))),u)).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(e(e(e(e(e(e(v,e(e(e(v,w),e(v6,w)),v6)),v7),v7),v8),e(v9,v8)),v9),u))).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(e(e(e(v,e(e(e(v,w),e(v6,w)),v6)),v7),v7),u))).
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(e(e(u,v),e(w,v)),w))).
P(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),v),e(x,v))).
P(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),w),e(v,w))),x)).
P(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),e(v7,v6)),v7))).
P(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),v7),e(v6,v7)))).
P(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),v8),v8)),v))),x)).
P(e(e(x,e(y,e(e(e(e(e(z,e(e(e(z,u),e(v,u)),v)),e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),y)),v8),e(v9,v8)),v9))),x)).
P(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),x)).
P(e(e(x,e(y,x)),y)).
P(e(x,e(e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),v6),e(v7,v6)),v7),x))).
P(e(x,e(e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x),v6),e(v7,v6)),v7))).
P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(e(v,w),e(v6,w)),v6)),x)),v7),e(v8,v7)),v8))).
P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x))).
P(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x))).
P(e(x,e(y,e(x,y)))).
% Following 30 sorted from temp.xcb.exp2.out3a purport to prove the remaining of the 15
% 7-symbol, including symmetry, excluding f7k.
P(e(e(e(e(e(e(x,y),x),y),z),u),e(z,u))).
P(e(e(e(e(e(x,e(y,x)),z),e(u,z)),u),y)).
P(e(e(e(e(e(x,e(y,y)),x),z),e(u,z)),u)).
P(e(e(e(e(e(x,e(y,y)),x),z),u),e(z,u))).
P(e(e(e(e(e(x,x),e(y,y)),z),e(u,z)),u)).
P(e(e(e(e(e(x,x),e(y,y)),z),u),e(z,u))).
P(e(e(e(e(x,e(y,e(x,y))),z),e(u,z)),u)).

```
P(e(e(e(e(x,e(y,e(x,y))),z),u),e(z,u))).
P(e(e(e(e(x,x),y),e(z,y)),z)).
P(e(e(e(e(x,y),e(z,y)),z),e(e(u,x),u))).
P(e(e(e(e(x,y),x),z),e(y,z))).
P(e(e(e(e(x,y),y),z),e(x,z))).
P(e(e(e(e(x,y),z),e(y,z)),x)).
P(e(e(e(x,e(y,x)),z),e(y,z))).
P(e(e(e(x,x),y),y)).
P(e(e(e(x,y),e(e(e(y,z),e(u,z)),u)),x)).
P(e(e(e(x,y),x),e(z,e(z,y)))).
P(e(e(e(x,y),y),x)).
P(e(e(x,e(x,y)),y)).
P(e(e(x,e(y,y)),x)).
P(e(e(x,x),e(e(e(y,z),y),z))).
P(e(e(x,x),e(y,y))).
P(e(e(x,y),e(e(e(z,x),z),y))).
P(e(e(x,y),e(y,x))).
P(e(x,e(e(e(y,z),y),e(x,z)))).
P(e(x,e(e(x,y),y))).
P(e(x,e(e(y,x),y))).
P(e(x,e(e(y,y),x))).
P(e(x,e(x,e(y,y)))).
P(e(x,e(y,e(y,x)))).
end_of_list.
```

A 71-Step Proof to Shorten

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Sun Jun  5 09:46:05 2005
The command was "otter".  The process ID is 19701.

-----> EMPTY CLAUSE at 468.57 sec ----> 110424 [hyper,2,6385,110057] $ANSWER(all_s_t_indep).

Length of proof is 71.  Level of proof is 23.

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

1 [] -P(e(x,y)) | -P(x) | P(y).
2 [] -P(e(e(a,b),e(b,a))) | -P(e(e(a,b),e(e(b,c),e(a,c)))) | $ANSWER(all_s_t_indep).
4 [] P(e(x,e(e(e(x,y),e(z,y)),z))).
170 [hyper,1,4,4] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v)).
171 [hyper,1,4,170] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),e(v6,w)),v6)).
172 [hyper,1,170,4] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v))).
174 [hyper,1,171,4] P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v6))).
175 [hyper,1,172,172] P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),x))).
176 [hyper,1,4,172] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v)),w),e(v6,w)),v6)).
178 [hyper,1,172,170] P(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w)),x)).
179 [hyper,1,172,4] P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w))).
180 [hyper,1,172,174] P(e(x,e(e(e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),v6),e(v7,v6)),v7),x))).
181 [hyper,1,4,174] P(e(e(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v6)),v7),e(v8,v7)),v8)).
183 [hyper,1,174,170] P(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),w),e(v,w))),x)).
193 [hyper,1,175,4] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),e(w,e(e(e(w,v6),e(v7,v6)),v7)))).
```

201 [hyper,1,176,4] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v)),w),v6),e(w,v6))).
202 [hyper,1,176,178] P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),e(v7,v6))),v7)).
206 [hyper,1,4,178] P(e(e(e(e(e(x,e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w)),x),v6),e(v7,v6)),v7)).
226 [hyper,1,172,179] P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(e(v,w),e(v6,w)),v6)),x)),v7),
e(v8,v7)),v8))).
229 [hyper,1,4,179] P(e(e(e(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w)),v6),
e(v7,v6)),v7)).
234 [hyper,1,179,174] P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(e(e(u,e(e(e(u,v),e(w,v)),w)),v6),
e(v7,v6)),v7),v8),v9),e(v8,v9))),v10),e(v11,v10)),v11)).
239 [hyper,1,179,4] P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(e(e(u,v),e(w,v)),w))),v6),e(v7,v6)),v7)).
253 [hyper,1,180,172] P(e(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),e(v6,w)),v6),
e(e(e(e(v7,e(e(e(v7,v8),e(v9,v8)),v9)),v10),v11),e(v10,v11)))).
264 [hyper,1,181,183] P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),e(e(e(w,e(e(e(w,v6),
e(v7,v6)),v7)),v8),v9)),e(v8,v9))).
385 [hyper,1,174,206] P(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),x)).
406 [hyper,1,170,226] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(u,e(e(e(u,v),e(w,v)),w)),e(e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)),v9))),v10),e(v9,v10))).
452 [hyper,1,201,229] P(e(e(x,e(y,e(e(e(e(e(z,e(e(e(z,u),e(v,u)),v)),e(e(w,e(e(e(w,v6),
e(v7,v6)),v7)),y)),v8),e(v9,v8)),v9))),x)).
474 [hyper,1,174,234] P(e(e(x,e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),e(e(e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)),v9),e(v10,v9))),v10)),x)).
509 [hyper,1,4,239] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(e(e(u,v),e(w,v)),w))),v6),
e(v7,v6)),v7),v8),e(v9,v8)),v9)).
517 [hyper,1,172,253] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v6)),
e(e(e(e(v7,e(e(e(v7,v8),e(v9,v8)),v9)),v10),v11),e(v10,v11)))).
580 [hyper,1,264,385] P(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x))).
605 [hyper,1,234,193] P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(e(e(e(v,e(e(e(v,w),e(v6,w)),v6)),v7),v7),u))).
711 [hyper,1,4,452] P(e(e(e(e(e(x,e(y,e(e(e(e(e(z,e(e(e(z,u),e(v,u)),v)),e(e(w,e(e(e(w,v6),
e(v7,v6)),v7)),y)),v8),e(v9,v8)),v9))),x),v10),e(v11,v10)),v11)).
805 [hyper,1,202,474] P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u)).
851 [hyper,1,509,517] P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(u,e(v,e(e(e(v,w),e(v6,w)),v6)))),u)).
934 [hyper,1,4,580] P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),v),e(w,v)),w)).
1108 [hyper,1,605,178] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(v,e(e(e(v,w),e(v6,w)),v6)))).
1196 [hyper,1,4,805] P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(w,v)),w)).
1242 [hyper,1,406,934] P(e(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),v),w),e(v,w)),e(v6,e(e(e(v6,v7),
e(v8,v7)),v8)))).
1264 [hyper,1,934,226] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(u,e(e(e(u,v),e(w,v)),w)),
e(e(v6,e(e(v7,e(e(e(v7,v8),e(v9,v8)),v9)),v6)),v10))),v11),e(v10,v11))).
1368 [hyper,1,4,1108] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(v,e(e(e(v,w),
e(v6,w)),v6))),v7),e(v8,v7)),v8)).
1484 [hyper,1,851,1196] P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),v),e(e(e(v,w),e(v6,w)),v6))).
1494 [hyper,1,201,1196] P(e(e(x,e(y,y)),x)).
1601 [hyper,1,406,1242] P(e(e(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),x)),e(v,e(e(e(v,w),
e(v6,w)),v6))),v7),v8),e(v7,v8)),e(v9,e(e(e(v9,v10),e(v11,v10)),v11)))).
1667 [hyper,1,851,1368] P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u),e(v,e(e(e(v,w),
e(v6,w)),v6))),v7),e(e(e(v7,v8),e(v9,v8)),v9))).
1810 [hyper,1,1484,1264] P(e(e(e(e(x,e(e(y,e(e(e(y,z),e(u,z)),u)),e(e(v,e(e(w,e(e(e(w,v6),
e(v7,v6)),v7)),v)),x))),v8),e(v9,v8)),v9)).
1902 [hyper,1,1494,1494] P(e(e(x,x),e(y,y))).
2288 [hyper,1,711,1601] P(e(e(e(x,e(y,e(e(e(y,z),e(u,z)),u))),v),e(x,v))).
2447 [hyper,1,1667,1810] P(e(e(e(e(e(e(x,y),x),y),z),e(u,z)),u)).
2637 [hyper,1,4,1902] P(e(e(e(e(e(x,x),e(y,y)),z),e(u,z)),u)).
2977 [hyper,1,264,2288] P(e(e(e(e(x,y),e(z,y)),z),x)).

3256 [hyper,1,805,2447] P(e(e(e(e(e(e(x,y),x),y),z),u),e(z,u))).
3352 [hyper,1,805,2637] P(e(e(e(e(e(x,x),e(y,y)),z),u),e(z,u))).
3639 [hyper,1,172,2977] P(e(e(e(e(x,y),z),e(y,z)),x)).
3982 [hyper,1,3256,2977] P(e(x,e(y,e(x,y)))).
4358 [hyper,1,2288,3639] P(e(e(e(x,y),e(e(e(y,z),e(u,z)),u)),x)).
4456 [hyper,1,4,3982] P(e(e(e(e(x,e(y,e(x,y))),z),e(u,z)),u)).
4613 [hyper,1,3256,4358] P(e(e(e(e(x,y),e(z,y)),z),e(e(u,x),u))).
4644 [hyper,1,4358,2977] P(e(e(e(e(e(x,y),e(z,y)),z),u),e(x,u))).
4667 [hyper,1,805,4456] P(e(e(e(e(x,e(y,e(x,y))),z),u),e(z,u))).
4741 [hyper,1,4358,4613] P(e(e(e(e(x,y),y),z),e(x,z))).
4745 [hyper,1,3352,4613] P(e(x,e(e(y,x),y))).
4853 [hyper,1,4667,4358] P(e(e(e(e(e(x,e(y,x)),z),e(u,z)),u),y)).
4979 [hyper,1,1494,4741] P(e(e(e(x,y),y),x)).
5410 [hyper,1,4979,4853] P(e(e(e(x,e(y,x)),z),e(y,z))).
5411 [hyper,1,4979,4613] P(e(e(x,y),e(e(e(z,x),z),y))).
5529 [hyper,1,5410,5411] P(e(x,e(e(e(y,z),y),e(x,z)))).
6015 [hyper,1,385,5529] P(e(e(e(e(x,y),x),z),e(y,z))).
6385 [hyper,1,4741,6015] P(e(e(x,y),e(y,x))).
6547 [hyper,1,4745,6385] P(e(e(x,e(e(y,z),e(z,y))),x)).
107512 [hyper,1,4741,4644] P(e(e(e(x,y),e(z,y)),e(x,z))).
108720 [hyper,1,6547,107512] P(e(e(e(x,y),z),e(e(y,x),z))).
108731 [hyper,1,6385,107512] P(e(e(x,y),e(e(x,z),e(y,z)))).
110057 [hyper,1,108720,108731] P(e(e(x,y),e(e(y,z),e(x,z)))).
110424 [hyper,2,6385,110057] $ANSWER(all_s_t_indep).

You now have in hand a 71-step proof that deduces, from *XCB*, a 2-basis for equivalential calculus (an axiom system consisting of two formulas, symmetry and transitivity). Can you find a shorter proof? You can use as targets the given 2-basis or one of the known single axioms; see the passive list of the given input file. A few years ago (April 3, 2005) I discovered a 22-step proof. Rather than giving the proof, which might indeed sharply reduce for you the excitement of seeking a shorter proof than length 71, I shall give some added background for this essay, in part relevant to finding the 22-step proof. You will learn, among other things, why I have chosen here in 2009 to produce this document.

In the mid-1980s, Woody Bledsoe asked me, in effect, to put this type of material on paper. In the mid-1990s, Robert Boyer echoed this sentiment independently, noting that nobody would ever go through my massive files. Ross Overbeek, in 2008, said such material should be available for those in the future. McCune is also in favor, noting that I might learn much as I attempt to capture on paper what is often hidden from me as I attack theorem after theorem. Indeed, often one does not know explicitly what one knows or what one does to win the game. I thank each of these four people for their encouragement.

In a workshop in the summer of 2004, Mark Stickel learned about a book I had written with Gail Pieper, *Automated Reasoning and the Discovery of Missing and Elegant Proofs*, Rinton Press. In Chapter 7 of that book, you will find many challenges and open questions. Among them is a challenge that asks for a proof of length less than 25 that shows *XCB* to be a single axiom. Yes, I had not yet found the cited 22-step proof, and I will shortly get to that. A few months after that workshop, Stickel sent me an answer to the question about the existence of such a proof, an affirmative answer. He had found, with one of his programs, a 24-step proof that answered the following specific question.

With condensed detachment as the only inference rule, and with the only admissible targets for the completion of a proof one of the fourteen shortest single axioms (other than *XCB*) or the independent 2-basis consisting of symmetry and transitivity, does there exist a proof of length strictly less than 25 that establishes *XCB* to be a single axiom for EC?

Stickel's result rekindled my interest in short proofs in the given context, and I found a 23-step proof on December 24, 2004. Some time later (I do not know precisely when, but I believe it was April 4, 2005), I found the cited 22-step proof.

I found that proof with some extensions of methodologies given in the cited book (that with Pieper). Because of the new advances in methodology (occurring in the preceding year) in general, because I had found with OTTER this charming 22-step proof, and because of the encouragement I had received, I decided to write this notebook. An additional motivating force was my discovery (with the new methodologies) of other shorter proofs, improving on what I had found about a year before. Some of these proofs are discussed in this notebook; others I intend to be the subject of later essays.

I am convinced that the use of the proof-refinement methodologies, especially those focusing on finding shorter proofs, can be used to replace the material in textbooks by simpler and easier-to-follow material. In particular, certain theorems are proved by relying on thought-to-be indispensable lemmas and assumed-to-be necessary inclusion of types of term. With OTTER, I have discovered proofs that avoid the use of such lemmas (in the Lukasiewicz infinite-value logic), and I have found proof after proof that avoids the use of a commonly occurring class of term (those of double negation). The experiments I detail may not be precisely those I conducted, for such information is lost to history. However, they will reflect the spirit of what I do and did. Moreover, in addition to providing templates for other studies, the details will permit you to independently reproduce, and thus verify, what is offered here.

Only unbelievable optimism would lead one to suggest that an algorithm exists for mining the treasure of mathematics and logic. The treasure in focus in this notebook takes the form of better and more elegant proofs, better in one or more measures. (If all goes as planned, this essay will be followed by others, each with its own emphasis.) In place of the mythical algorithm, I will present methodologies and strategies whose use often sharply increases the likelihood of success. Sprinkled throughout will be commentary that reflects my excitement and that is intended to produce excitement for the reader, ideas whose pursuit by someone could result in a rewarding publication, and detail and data whose analysis may enable the enhancing of automated reasoning programs so that they can provide even greater assistance in solving numerous mysteries.

Yes—as my friend Overbeek suggests that I point out—anyone who reads this essay, or any succeeding essay in this series, is more than welcome to take an idea that is offered and extend it, modify it, and in general use it for enjoyment or for research. Challenge problems and open questions will be included. Despite the numerous and significant results that have been obtained with an automated reasoning program—McCune's OTTER is the program featured—a precise analysis and thorough understanding of how everything works are still lacking. From a different viewpoint, the nature of proof and of proof search still resists the efforts of fine minds.

Although the focus in this notebook is on logic and, indirectly, on mathematics, the notions and observations must (it seems to me) apply to reasoning whose goal is other than a formal proof of some purported or actual theorem. I am firmly convinced of the accuracy of this remark despite the sometimes-uttered objection that real-world problems, such as those concerned with queries of large databases, involve large sets of axioms in contrast to the small sets of axioms to be studied here.

One of the delightful aspects of automated reasoning, at least when OTTER is in use, is that you can obtain proofs that may charm both you and others. Further, you need not be an expert to reap such rewards. This study (of the *BCSK* logic) illustrates this observation: I knew essentially nothing about the area, and I still know almost nothing regarding its nature and theory. Nevertheless, with indispensable aid from OTTER, new and beautiful proofs were found and marvelous theorems proved, theorems perhaps never before considered.

With the preceding as stage setting and with this small glimpse of my intentions, I now turn to the study of concern. I again note that the commentary is intended to provide one or more clues about how things work, how the goals are reached, and which actions are more likely to lead to success and, eventually, to a fuller understanding of the nature of proof and of proof search. The key word is "intended"; indeed, at this time, I cannot be certain that my comments are accurate; I make conjectures that may, in fact, not hold. However, a reading of what follows will, for some, lead to intrigue and excitement and, certainly important, an ability to discover proofs of interest to the worlds of mathematics and logic. I shall include input files to enable you to verify the claims I make and to begin an independent study of some type. I also shall include ideas for future—or even better, present—research. I am fairly certain that the experiments I present at least resemble the original ones that led to most satisfying discoveries. They do capture the spirit of what I do, but in many cases they are not identical (many of the actual experiments are simply lost to history).

2. The Field in Focus and Some Background

This story begins in the middle of a longer story, a story that commenced with a request by Spinks to find "short" proofs. The area of interest is known as the *BCSK* logic (whose axioms I give almost immediately), an extension of which is related to classical propositional calculus. I was unfamiliar with that logic, but I was intrigued by the problem, which involved the following nine axioms (in clause notation), where the functions *i* and *j* respectively denote *strong* and *weak implication*.

```
P(i(x,i(y,x))).                    % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).     % (A2)
% P(i(i(i(x,y),x),x)).              % (A3)
P(i(x,j(y,x))).                    % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).     % (A5)
% P(i(j(x,j(y,z)),j(y,j(x,z)))).       % (A6)
% P(i(j(j(x,y),x),x)).              % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).         % (A8)
P(j(i(x,y),j(x,y))).               % (A9)
```

As those familiar with OTTER know, by placing a percent sign in column 1, three of the axioms are "commented out"; that is, *A*3, *A*6, and *A*7 are prevented from participation in the study that is central to this essay. You will soon see why these three are commented out.

Condensed detachment is the inference rule in use here; therefore, because of the presence of two types of implication, two 3-literal clauses (the following) are present, as well as the use of hyperresolution.

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

Veroff supplied me with various proofs, many or all obtained by using his powerful *sketches* technique. As I sought short proofs for two targets, I found—to my utter surprise and to that of Spinks—that two axioms were dependent (*A*3 and *A*6) on the remaining seven and, regarding *A*7, that it was not needed for the proofs I sought. You thus see why the three are commented out. Spinks was especially pleased to find that *A*7 is not needed for the proofs of dependency. After that find, all experiments were conducted with axioms 1, 2, 4, 5, 8, and 9, which suggests (from the viewpoint of logic or mathematics) that a weaker *variety* might merit study, a variety axiomatized with the cited six axioms. The proof that OTTER completed for the dependence of axiom *A*3 has length 14 and level 10. That proof, the following, provided the impetus for what is reported here. (The length of proofs included here does not distinguish between the two (so to speak) forms of condensed detachment, one for the function *i* and one for *j*, and it measures just the number of deduced steps.)

A 14-Step Proof

```
----- Otter 3.3d, April 2004 -----
The process was started by wos on jaguar.mcs.anl.gov,
Thu May 27 10:43:03 2004
The command was "otter".  The process ID is 31886.
----> UNIT CONFLICT at   0.02 sec ----> 150 [binary,149.1,17.1] $ANS(a3).

Length of proof is 14.  Level of proof is 10.

---------------- PROOF ----------------
```

```
6 [] -P(i(x,y)) | -P(x) | P(y).
7 [] -P(j(x,y)) | -P(x) | P(y).
9 [] P(i(x,i(y,x))).
10 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
11 [] P(i(x,j(y,x))).
12 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
13 [] P(i(j(i(x,y),y),j(i(y,x),x))).
14 [] P(j(i(x,y),j(x,y))).
17 [] -P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
24 [hyper,6,9,9] P(i(x,i(y,i(z,y)))).
38 [hyper,6,10,10] P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
40 [hyper,6,10,24] P(i(i(x,y),i(x,i(z,y)))).
41 [hyper,6,10,9] P(i(i(x,y),i(x,x))).
46 [hyper,7,14,40] P(j(i(x,y),i(x,i(z,y)))).
58 [hyper,6,38,41] P(i(i(x,i(x,y)),i(x,y))).
97 [hyper,7,14,58] P(j(i(x,i(x,y)),i(x,y))).
115 [hyper,6,13,97] P(j(i(i(x,y),x),x)).
120 [hyper,6,11,115] P(j(x,j(i(i(y,z),y),y))).
124 [hyper,6,12,120] P(j(j(x,i(i(y,z),y)),j(x,y))).
129 [hyper,7,124,46] P(j(i(i(i(x,y),z),y),i(x,y))).
138 [hyper,7,124,129] P(j(i(i(i(i(x,y),x),z),x),x)).
144 [hyper,6,13,138] P(j(i(x,i(i(i(x,y),x),z)),i(i(i(x,y),x),z))).
149 [hyper,7,144,9] P(i(i(i(x,y),x),x)).
```

3. The First Experiment

The year following OTTER's finding the 14-step proof witnessed some refinements of various methodologies offered in my book (with G. W. Pieper) *Automated Reasoning and the Discovery of Missing and Elegant Proofs*, Rinton Press. Most satisfying, I had found an approach that culminated in the discovery of a 22-step proof showing *XCB* to be a single axiom—a proof three steps shorter than that featured in the open question cited in Section 1. With this motivation, and because of my interest (as well as that of Spinks) in short proofs, I decided to see whether a proof of length strictly less than 14 could be found establishing *A*3 to be dependent on axioms 1, 2, 4, 5, 8, and 9.

The following input file, which relies on the given 14-step proof, captures the approach (or one very similar) taken in the first experiment. (The inclusion throughout of appropriate input files provides a means to independently check the results that are given and also enables you to more easily attack problems of some other type.)

First Input File

```
set(hyper_res).
assign(max_weight,23).
assign(change_limit_after,400).
assign(new_max_weight,16).
clear(print_kept).
% clear(for_sub).
set(ancestor_subsume).
set(back_sub).
% clear(set_sub).
assign(max_mem,600000).
%  assign(max_seconds,7).
% set(control_memory).
% assign(report,900).
%  assign(pick_given_ratio,4).
```

```
assign(max_proofs,-1).
%set(order_history).
%set(input_s_first).
set(s_queue).
%set(print_level).
set(order_history).
assign(max_distinct_vars,4).
assign(heat,0).

weight_list(pick_and_purge).
%  Following 14 from temp.spinks1.depax37.out1e prove A3 dependent on 1 2 4 5 8 9.
weight(P(i(x,i(y,i(z,y)))),1).
weight(P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))),1).
weight(P(i(i(x,y),i(x,i(z,y)))),1).
weight(P(i(i(x,y),i(x,x))),1).
weight(P(j(i(x,y),i(x,i(z,y)))),1).
weight(P(i(i(x,i(x,y)),i(x,y))),1).
weight(P(j(i(x,i(x,y)),i(x,y))),1).
weight(P(j(i(i(x,y),x),x)),1).
weight(P(j(x,j(i(i(y,z),y),y))),1).
weight(P(j(j(x,i(i(y,z),y)),j(x,y))),1).
weight(P(j(i(i(i(x,y),z),y),i(x,y))),1).
weight(P(j(i(i(i(i(x,y),x),z),x),x)),1).
weight(P(j(i(x,i(i(i(x,y),x),z)),i(i(i(x,y),x),z))),1).
weight(P(i(i(i(x,y),x),x)),1).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(j(x,y)) | -P(x) | P(y).
%  -P(i(i(A,B),j(A,B))) |
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).        % Lemmas
end_of_list.

list(sos).
% Axioms
P(i(x,i(y,x))).                          % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).   % (A2)
%  P(i(i(i(x,y),x),x)).                   % (A3)
P(i(x,j(y,x))).                          % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).   % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).        % (A6)
%  P(i(j(j(x,y),x),x)).                   % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).        % (A8)
P(j(i(x,y),j(x,y))).                   % (A9)
end_of_list.

list(passive).
-P(i(j(a1,j(a2,a3)),j(a2,j(a1,a3)))) | $ANS(A6).
-P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
-P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
-P(i(a1,j(a2,a1))) | $ANS(a4).
-P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).                   % Lemma
```

```
-P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).      % Lemma
-P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).      % Lemma
end_of_list.

list(demodulators).
% (P(i(i(i(x,y),x),x)) = junk).  %  A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk).  %  A6
(P(i(j(j(x,y),x),x)) = junk).  %  A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

list(hot).
-P(i(x,y)) | -P(x) | P(y).             % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

I recommend a glass of wine and some patience for the following promised discourse accompanied by commentary; but, at least for some, its reading will prove interesting in part because of dispelling an understandable myth. Since the command set(sos_queue) was used to have OTTER employ a breadth-first search and since that type of search is so often impractical, a rather lengthy discourse is in order. (Far less detail is given for later experiments.) A glance at the input file reveals OTTER's wealth of choices, some regarding options, some regarding parameter values, and some regarding lists. How in the world does one make wise choices in the context of the three facets? Before you conjecture that an algorithm exists for making good choices, I note that I know of none; indeed, I believe that none currently exists. However, no doubt because of the many thousands of experiments with OTTER, I often do make effective initial choices and effective changes as a series of experiments proceeds. With this notebook and others I plan to write, my goal is to provide commentary and examples of input files that will enable other researchers to benefit from my years with automated reasoning and will sharply reduce the time to learn, perhaps by osmosis, how to attack a question or problem profitably.

As I began my study, two conflicting forces merit focus. To understand this conflict, one might consider, for example, the value that can be assigned to max_distinct_vars. When the value k is assigned, OTTER will discard any deduced conclusion whose number of distinct variables strictly exceeds k. If the value is, say, 3, and an equation or formula is deduced that relies on x, y, z, and u, and possibly additional variables, that item will be discarded. One force suggests that generosity is the choice: Give the program much room in which to operate, and therefore assign a high value to max_distinct_vars. Because of the hugeness of the typical space of conclusions that can be drawn, the second force suggests that a small value be assigned to prevent the program from drowning. Indeed, too small a value can prevent OTTER from finding any proof because *no* proof exists within the constraint in use; too large a value can prevent it from completing a proof because of getting lost in the vast number of conclusions to be considered.

Three experiments nicely show what can happen when the value assigned to max_distinct_vars is respectively 3, 4, and 5, and when the search is based on breadth first or level saturation, and when all other options and parameters are otherwise alike in the three experiments. (For clarity, by definition, level-0 clauses are those that are input, level-1 are those that are deduced from applying the inference rules to level-0 clauses, and level-k clauses are those that are deduced with the property that one of the parents has level k-1.) The theorem to prove is that which establishes $A3$ to be dependent on 1, 2, 4, 5, 8, and 9. The respective input files include that previously given. For levels 1 through 7, the numbers of the last clause retained when 3 is assigned to max_distinct_vars are, respectively, 41, 79, 174, 424, 1150, 2289, and 3824. When 4 is the assigned value, they are 45, 114, 364, 1526, 5007, 14384, and 42279. When 5 is the assigned value, they are 45, 122, 488, 3069, 9838, 31886, and 84361. A glance at the three sets of numbers shows what the trend is and correctly indicates what would occur at higher and still higher levels for the respective three assignments to max_distinct_vars.

Now you can contemplate the awesome set of numbers when the value assigned is 26, which was the number of distinct variables present in Steve Winker's first proof (one of length 159) that the formula *XHN* is a single axiom for equivalential calculus. For one more data item that permits a glimpse of the horror that awaits an automated reasoning program applying a breadth-first search, I have in hand a proof of level 227 for a proof of a theorem in loop theory. Yes, in many cases, a breadth-first search is out of the question.

Occasionally, however, such a search proves useful. Indeed, the given input file produced such a result, although its usefulness was not obvious at the time. You might therefore conjecture that such a search might be a good way for seeking the shortest proof of a given theorem. Although, if the combinatoric explosion does not occur, a level-saturation search is useful when seeking a minimal-level proof of a theorem, as the following shows, the conjecture does not hold. Indeed, one can imagine that the theorem to prove admits many proofs. Let the theorem to be proved be denoted by (the clause) *C*. If a first proof consists of deducing *C* from applying, say, condensed detachment to *A* and *B* with the respective subproofs of the two each of length 4 and level 4, then the length of the this total proof of *C* is 9, and its level is 5. If a second proof consists of *E*, *f*, *G*, *H*, *J*, and *C*, where *E* is obtained by applying condensed detachment to two (input) axioms and each of *F* through *C* is obtained from its predecessor (in the given list), then this second proof has level 6 and, more important, length 6. You can now see, with the given example, that a level-saturation search, even if a proof is found, may not return a minimal-length proof although it may return a minimal-level proof. Further, if the first proof that can be found has level *k* equal, say, to 16, and if there exists a shorter proof at level, say, 19, the rapid growth of the size of the levels might well prevent the discovery of the second proof.

With the treatment of the command set(sos_queue), found in the given input file, now in hand, the other items of that file merit examination. I begin with the less interesting (at least for this first experiment) items, the "set" commands. In order, set(hyper_res) informs the program that hyperresolution is the inference rule to be used, chosen because the study concerns condensed detachment. Next, the inclusion of the command set(ancestor_subsume) tells the program to seek shorter proofs (preferring the strictly shorter derivation when two paths lead to the same conclusion). I always include this command when seeking shorter proofs; its inclusion does slow the program, but the command is effective in the given context. The inclusion of the command set(back_sub) has the program apply back subsumption, to purge already-retained conclusions if and when a newly deduced conclusion is strictly more general. This command is always to be included when ancestor subsumption is in use. Finally, the inclusion of the command set(order_history) causes the program to list (by number of the clause), in order, for each retained conclusion the nucleus of the hyperresolution, the major premiss, and the minor premiss. (Yes, like Church, I clearly prefer this spelling of *premiss* despite the now-current alternative, premise.) In studies concerned with the use of condensed detachment, one occasionally wishes to quickly identify which formula is used as the major premiss and which as the minor. Indeed, a possibly useful strategy would have the program limit which formulas are to be used as major and which as minor premiss. My colleague Ulrich has shown interest in such a strategy, and I offer the topic to you as a possible area for research.

In contrast to the "set" statements, the justifications for the "assign" statements with their assigned values found in the First Input File are less precise, based more on years of experience. The less significant are those concerning memory and heat. The assign(max_mem,600000) limits OTTER to the use of 600 megabytes of memory, and the assign(heat,0) has the program avoid the use of the hot list strategy. A bit more interesting is assign(max_proofs,-1), encouraging the program to find as many proofs as time and memory permit. Indeed, typically OTTER will find a number of proofs of each of the targets included in the input; they are *not* always descending in length (as will be discussed later). The statement assign(max_distinct_vars) is of interest and proved crucial in the first experiment. In particular, the 14-step proof of the dependency of *A*3 relies on but three distinct variables. My intention was to give the program more latitude in the hope of finding a proof of length strictly less than 14. Although such did not occur, as you will see, a valuable proof was nevertheless found. Finally, the following three commands, taken together, merit a bit of comment.

assign(max_weight,23).
assign(change_limit_after,400).
assign(new_max_weight,16).

In symbol count, the longest formula in the 14-step proof has a weight (number of symbols) of 20. In part because of that observation and in part because of the assignment of 4 to max_distinct_vars, to give the program some room in which to operate, the value 23 was assigned to max_weight. (With the value *k* assigned to max_weight, the program will discard any deduced clause whose weight, usually measured in symbol count, strictly exceeds *k*; as you will see, symbol count can be replaced.) In answer to a question that might arise, I guessed that a larger value might prevent the program from completing the exploration of enough levels; the level of the 14-step proof is 10. A smaller value might block the finding of an interesting proof or block all proofs. But, in that OTTER might be forced to explore farther than level 10, I included the second of the three commands, instructing the program to lower the max_weight after 400 clauses were chosen to initiate the use of condensed detachment. The third command reduced the max_weight to 16, causing the size of levels to grow less rapidly than it would otherwise, but still giving some room.

Next in order is a discussion of lists as they are used in the First Input File. The first list, pick_and_purge, is where the 14-step proof comes into play. Each of the 14 formulas is placed as a *resonator* in the weight_list(pick_and_purge), a list whose use (in this experiment) is to guide OTTER toward a proof of interest. A resonator is a formula or equation whose functional pattern is the key, with all variables treated as indistinguishable, as simply marking that a variable appears in the corresponding spot. Any deduced item that matches a resonator is assigned the value assigned to the matching resonator. The smaller the value, the higher the priority for directing a program's reasoning. In this first experiment, any formula that matches one of the fourteen steps of the proof showing *A*3 to be dependent is given preference by OTTER for initiating an application of condensed detachment. I must be totally clear: Not only if and when one of the fourteen steps is deduced is it given high priority for driving the reasoning, but any formula that has a functional pattern that agrees with one of the fourteen is also given such preference. Indeed, the third step of the 14-step proof is matched at the resonator level, but not identically, by the 88th clause in the output, the 88th that is used to initiate applications of condensed detachment. Here are the two formulas.

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

The second of the two formulas is in fact not a formula in the 14-step proof. But, because of its functional shape (with all variables treated as indistinguishable), it is assigned the value 1, which is the value assigned to the resonator that corresponds to the first of the two formulas. If a breadth-first search had not been in use, and if all of the input clauses (from the set-of-support list) were chosen first to initiate inference rule application—such clauses are called given clauses—then the second formula would have been chosen as the thirteenth given clause even though it was clause (65). (If you wish to see precisely what occurs, you can simply take the given input file, comment out the sos_queue command by placing % in column 1, and remove the % from the set(input_sos_first) command.) In other words, a number of already-retained clauses that had not yet been chosen as given clauses would have been delayed for consideration because the clause corresponding to the second formula had been assigned a weight (value) of 1. Therefore, when that formula was deduced and retained—and was clause (125)—it was chosen in order because of the command set(sos_queue). Summing up, were it not for the use of a breadth-first (level-saturation) approach, the given fourteen resonators would have had a big effect.

You might again ask about the choice of sos_queue. My intention was to find a shorter proof than length 14, if one existed, and sometimes level saturation works in that context. One reason is that, with level saturation, clauses are considered even though they have high weight (low priority) for driving the reasoning. Indeed, their weight or assigned value is ignored with level saturation; this contrasts to using the weight, which causes the program to prefer clauses with small weight. For example, a clause with weight 20, when level saturation is not in use, might wait a long time—perhaps forever—before being chosen as a given clause. Therefore, if a shorter proof existed and required the use of a rather long formula, level saturation might unearth this fact.

Of course, any long clause that matches a resonator in the input will not be treated as a long clause if the assigned value to the resonator is a small number. However, if a long clause can be used to complete a shorter proof and if that clause does not match a resonator, all being equal, its weight will be determined solely by symbol count. In the proof that was found (the following), one of length 17 and level 7, nine clauses occur that are not in the 14-step proof, at least one of which might have never been considered, or perhaps not for a long time.

A 17-Step Proof

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on lemma.mcs.anl.gov,
Thu May 19 19:22:28 2005
The command was "otter". The process ID is 28933.
----> UNIT CONFLICT at 204.37 sec ----> 27442 [binary,27441.1,12.1] $ANS(a3).

Length of proof is 17. Level of proof is 7.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(j(x,y)) | -P(x) | P(y).
4 [] P(i(x,i(y,x))).
5 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
6 [] P(i(x,j(y,x))).
8 [] P(i(j(i(x,y),y),j(i(y,x),x))).
9 [] P(j(i(x,y),j(x,y))).
12 [] -P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
26 [hyper,1,4,4] P(i(x,i(y,i(z,y)))).
27 [hyper,1,5,5] P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
29 [hyper,1,5,4] P(i(i(x,y),i(x,x))).
47 [hyper,1,5,26] P(i(i(x,y),i(x,i(z,y)))).
52 [hyper,1,27,26] P(i(i(x,i(i(y,x),z)),i(x,z))).
56 [hyper,1,27,29] P(i(i(x,i(x,y)),i(x,y))).
60 [hyper,1,29,6] P(i(x,x)).
144 [hyper,1,4,52] P(i(x,i(i(y,i(i(z,y),u)),i(y,u)))).
163 [hyper,2,9,56] P(j(i(x,i(x,y)),i(x,y))).
190 [hyper,1,5,60] P(i(i(i(x,y),x),i(i(x,y),y))).
502 [hyper,1,52,144] P(i(i(i(x,y),z),i(y,z))).
595 [hyper,1,8,163] P(j(i(i(x,y),x),x)).
723 [hyper,1,4,190] P(i(x,i(i(i(y,z),y),i(i(y,z),z)))).
1904 [hyper,1,47,502] P(i(i(i(x,y),z),i(u,i(y,z)))).
3317 [hyper,1,5,723] P(i(i(x,i(i(y,z),y)),i(x,i(i(y,z),z)))).
9553 [hyper,1,3317,1904] P(i(i(i(x,y),z),i(i(i(y,z),u),u))).
27441 [hyper,2,595,9553] P(i(i(i(x,y),x),x)).
```

If you are curious about how important the inclusion of the fourteen resonators was, you can take the First Input File and comment out the entire weight_list(pick_and_purge), which will cause all formulas to be treated based solely on symbol count. You might also independently see whether a shorter proof than length 14 exists showing *A*3 is dependent on 1, 2, 4, 5, 8, and 9; I suspect that no shorter proof does exist, but I have not proved that such is the case.

As for the other lists in the First Input File, the usable list contains the 3-literal clauses for the two condensed-detachment nuclei, one for the function *i* and one for *j*. The list(demodulators) lays the groundwork for seeking shorter proofs by blocking steps of a proof, usually one at a time. The hot list is not used here because heat is assigned the value 0. The list(sos) is the place you look to to see which items in the beginning are used to initiate applications of the inference rules in use, in this case, hyperresolution. I placed all of the axioms to be used—1, 2, 4, 5, 8, and 9—in that list. Ordinarily, as one familiar with the set of support strategy knows, list(sos) is where you place the added hypothesis (special hypothesis) of the study and, possibly, the denial of the conclusion. In the case in focus, no special hypothesis exists. If you enjoy unexplored areas, you might try proving that *A*3 is dependent with a variation on the First Input File: namely, move some of the axioms to list(usable), and see what occurs. I have not tried doing so. Possibly, a quite different proof will be found. More generally, for any input file given in this notebook or later notebooks, a different list(sos) might prove at least amusing.

Finally, list(passive) is used to test for unit conflict, the typical test for assignment completion such as the completion of a proof, and for forward subsumption. I also use that list to monitor progress. For example, if I am seeking a proof of a conjunction, I place the negation of the conjunction in list(usable) but place each of its negated members in list(passive). I thus obtain, if successful, proofs of the individual members of the conjunction. (Those proofs sometimes are useful in the application of the *cramming strategy*, a topic for later.) I also often place, in list(passive), the negations of the proof steps of a proof, sometimes with the intention in the future of keying on the various subproofs, a topic at least for later notebooks.

The preceding lengthy material finishes my treatment of the first experiment, that focusing on finding a proof of length strictly less than 14 of the dependence of *A*3. Although that goal was not reached, as will become clear shortly as the second experiment takes center stage—now—the new proof of length 17 turned out to be most useful. By the way, did you notice that the assignment of 4 to max_distinct_vars played a key role for the 17-step proof? Yes, three of its deduced steps rely on four distinct variables. I suspect that no proof exists of level strictly less than 7, that of the 17-step proof.

4. The Second Experiment and Related Experiments

At this point, you might at first find somewhat jarring the change of emphasis, of goal. Indeed, as is so typical of my research, I was sufficiently delighted with this 17-step proof (of the dependency of *A*3) of substantially lower level, 7 versus 10, that I decided to seek a proof shorter than I had found that deduced, from 1, 2, 4, 5, 8, and 9, the following.

```
P(i(i(x,y),j(x,y))).  %  thesis 1
```

The deduction of this formula, so I was informed by Spinks, was one of two important results he sought. I had in hand proofs of each from Veroff—the second theorem is the following in its negated form—and had found, perhaps a year ago, a proof of length 27 of the preceding formula (from 1, 2, 4, 5, 8, and 9).

```
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).
```

(This second theorem will be, if all goes as planned, the central figure in a later essay.) Here is the 27-step proof for the first target, thesis 1.

A 27-Step Proof of Thesis 1

```
----- Otter 3.3d, April 2004 -----
The process was started by wos on lemma.mcs.anl.gov,
Thu May 27 10:44:48 2004
The command was "otter".  The process ID is 14323.
----> UNIT CONFLICT at   0.33 sec ----> 1392 [binary,1391.1,19.1] $ANS(THESIS_1).

Length of proof is 27.  Level of proof is 13.

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

6 [] -P(i(x,y)) | -P(x) | P(y).
7 [] -P(j(x,y)) | -P(x) | P(y).
9 [] P(i(x,i(y,x))).
10 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
```

```
11 [] P(i(x,j(y,x))).
12 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
13 [] P(i(j(i(x,y),y),j(i(y,x),x))).
14 [] P(j(i(x,y),j(x,y))).
19 [] -P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).
24 [hyper,6,9,9] P(i(x,i(y,i(z,y)))).
27 [hyper,6,9,11] P(i(x,i(y,j(z,y)))).
35 [hyper,6,10,10] P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
37 [hyper,6,10,27] P(i(i(x,y),i(x,j(z,y)))).
39 [hyper,6,10,9] P(i(i(x,y),i(x,x))).
43 [hyper,6,35,24] P(i(i(x,i(i(y,x),z)),i(x,z))).
46 [hyper,6,10,37] P(i(i(i(x,y),x),i(i(x,y),j(z,y)))).
59 [hyper,6,39,11] P(i(x,x)).
63 [hyper,6,9,43] P(i(x,i(i(y,i(i(z,y),u)),i(y,u)))).
72 [hyper,6,9,46] P(i(x,i(i(i(y,z),y),i(i(y,z),j(u,z))))).
83 [hyper,6,43,63] P(i(i(i(x,y),z),i(y,z))).
89 [hyper,6,37,13] P(i(j(i(x,y),y),j(z,j(i(y,x),x)))).
122 [hyper,6,43,72] P(i(x,i(i(x,y),j(z,y)))).
130 [hyper,7,14,59] P(j(x,x)).
137 [hyper,7,14,12] P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
146 [hyper,7,14,83] P(j(i(i(x,y),z),i(y,z))).
224 [hyper,6,11,130] P(j(x,j(y,y))).
252 [hyper,7,137,137] P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))).
478 [hyper,7,252,224] P(j(j(x,j(x,y)),j(x,y))).
625 [hyper,6,11,478] P(j(x,j(j(y,j(y,z)),j(y,z)))).
745 [hyper,7,137,625] P(j(j(x,j(y,j(y,z))),j(x,j(y,z)))).
958 [hyper,7,745,14] P(j(i(x,j(x,y)),j(x,y))).
1098 [hyper,6,89,958] P(j(x,j(i(j(y,z),y),y))).
1166 [hyper,7,137,1098] P(j(j(x,i(j(y,z),y)),j(x,y))).
1246 [hyper,7,1166,146] P(j(i(i(x,j(y,z)),y),y)).
1328 [hyper,6,13,1246] P(j(i(x,i(y,j(x,z))),i(y,j(x,z)))).
1391 [hyper,7,1328,122] P(i(i(x,y),j(x,y))).
```

The target in this second experiment, and possibly others to follow, was a proof of thesis 1 of length strictly less than 27 (applications of condensed detachment). In that I had spent much time and effort and relied on a variety of methodologies, many found in the book titled *Automated Reasoning and the Discovery of Missing and Elegant Proofs*, my seeking of a shorter proof might on the surface seem to be at best rather odd. As a reminder, because of the presence of two types of implication, two 3-literal clauses are being used. In the following input file, you will see how the first experiment came into play, specifically, its success.

Second Input File

```
set(hyper_res).
assign(max_weight,23).
%  assign(change_limit_after,400).
%  assign(new_max_weight,16).
clear(print_kept).
% clear(for_sub).
set(ancestor_subsume).
set(back_sub).
% clear(set_sub).
assign(max_mem,600000).
%  assign(max_seconds,2).
```

```
% set(control_memory).
% assign(report,900).
assign(pick_given_ratio,4).
assign(max_proofs,-1).
%set(order_history).
%set(input_sos_first).
%  set(sos_queue).
%set(print_level).
set(order_history).
assign(max_distinct_vars,3).
assign(heat,0).

weight_list(pick_and_purge).
%  weight(i(i(i(i($(1),$(1)),$(1)),$(1)),$(1)),100).
%  weight(j(j(j(j($(1),$(1)),$(1)),$(1)),$(1)),100).
%  Following 17 prove A3 dependent, smaller basis, temp.spinks2.a3.out1d1, 6 in the 14 not in this 17.
weight(P(i(x,i(y,i(z,y)))),0).
weight(P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))),0).
weight(P(i(i(x,y),i(x,x))),0).
weight(P(i(i(x,y),i(x,i(z,y)))),0).
weight(P(i(i(x,i(i(y,x),z)),i(x,z))),0).
weight(P(i(i(x,i(x,y)),i(x,y))),0).
weight(P(i(x,x)),0).
weight(P(i(x,i(i(y,i(i(z,y),u)),i(y,u)))),0).
weight(P(j(i(x,i(x,y)),i(x,y))),0).
weight(P(i(i(i(x,y),x),i(i(x,y),y))),0).
weight(P(i(i(i(x,y),z),i(y,z))),0).
weight(P(j(i(i(x,y),x),x)),0).
weight(P(i(x,i(i(i(y,z),y),i(i(y,z),z)))),0).
weight(P(i(i(i(x,y),z),i(u,i(y,z)))),0).
weight(P(i(i(x,i(i(y,z),y)),i(x,i(i(y,z),z)))),0).
weight(P(i(i(i(x,y),z),i(i(i(y,z),u),u))),0).
weight(P(i(i(i(x,y),x),x)),0).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).              % Modus
-P(j(x,y)) | -P(x) | P(y).              % Modus
%  -P(i(i(A,B),j(A,B))) |
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).
end_of_list.

list(sos).
%  Axioms
P(i(x,i(y,x))).                        % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).   % (A2)
%  P(i(i(i(x,y),x),x)).                  % (A3)
P(i(x,j(y,x))).                        % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).   % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).        % (A6)
%  P(i(j(j(x,y),x),x)).                  % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).        % (A8)
P(j(i(x,y),j(x,y))).                   % (A9)
```

```
end_of_list.

list(passive).
-P(i(j(a1,j(a2,a3)),j(a2,j(a1,a3)))) | $ANS(A6).
-P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
-P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
-P(i(a1,j(a2,a1))) | $ANS(a4).
-P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).            % Lemma
-P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).   % Lemma
-P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).   % Lemma
end_of_list.

list(demodulators).
%  (P(i(i(i(x,y),x),x)) = junk).  %  A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk).  %  A6
(P(i(j(j(x,y),x),x)) = junk).  %  A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

list(hot).
-P(i(x,y)) | -P(x) | P(y).          % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

A quick glance at this Second Input File reveals how the results of the first experiment influenced this one. Specifically, in the weight_list(pick_and_purge), you find seventeen weight templates, each as a resonator corresponding to one of the 17 steps of the proof obtained in the first experiment. Before those seventeen templates, you find two templates that are commented out; they are used eventually to increase the effectiveness of OTTER by enabling the program to discard four immediately nested occurrences of *i* and four immediately nested occurrences of *j*. I included those seventeen resonators because they correspond to a proof quite different from the 14-step proof I had found months and months ago, which led me to conjecture that their presence might lead to a proof quite unlike the 27-step proof just given. Yes, of course, the two targets apparently have little or nothing to do with each other, in the first experiment, *A*3, and in the second, thesis 1. You therefore may be justifiably skeptical about using these resonators. But that is how I do my research, often using resonators from a proof of one theorem to influence the seeking of a proof of another.

To push the program further in the direction of finding a proof somewhat or even very different from the 27-step proof, I assigned the value 3 to max_distinct_vars. Do you see why this assignment of 3 might force OTTER to find a different proof? As a clue, what is it about the given 27-step proof that you can use to force OTTER to find a different proof, perhaps sharply different? In addition to its length, is there an obvious property of the proof? Clearly, I am encouraging you to look at the proof, not in depth, but at its attributes; my aim is to encourage you to participate in the intrigue of this type of problem. Such examinations might, perhaps not too long from now, lead to a fuller understanding of proof and of the space of conclusions to be searched.

Well, one of the aspects of the 27-step proof, as well as that of any proof, is its variable richness. Whereas the *complexity* of a proof is defined as the longest formula or equation among its deduced steps, the *variable richness* is defined as the maximum number of distinct variables present in at least one of its deduced steps. Rather than the boring and sometimes consuming task of computing the complexity of a proof, you can far more easily determine its variable richness. For the 27-step proof, its variable richness is

4; two of its steps rely on four distinct variables. Therefore—and you are now beginning to get a taste of research—an assignment of a value strictly less than 4 to max_distinct_vars will force OTTER to attempt to complete a different proof. And you see why I chose the value 3; I did not try the value 2 because my experience says, in cases of the type under discussion, 2 never works.

Another item you might wonder about is assign(pick_given_ratio,4). Ordinarily, for initiating inference-rule application, OTTER chooses from the available items that based on weight. As noted, the weight of a formula or equation is its symbol count unless otherwise specified. The "otherwise" occurs when some weight template dictates a weight for the item. You can, for example, have the program give a weight of 1 to a formula whose symbol count is, say, 23. To do this, you merely take the item and place it, with an assigned value of 1, in, say, weight_list(pick_and_purge). In contrast to choosing given clauses (those that initiate a path of reasoning) based on complexity, as discussed, the program can be instructed to conduct a level-saturation approach. The value assigned to the pick_given_ratio blends the two; if the value is 4, then four clauses are chosen by complexity (weight), 1 by first come first serve, then 4, and 1, and so on. Now for the results, with some analysis and commentary, of what occurred in the second experiment.

The first two proofs (of seven) OTTER completed each showed *A*3 to be dependent on the usual (1, 2, 4, 5, 8, and 9), of respective lengths 17 and 14; in other words, they were not proofs of thesis 1. (Now you have in hand the use and value of assigning -1 to max_proofs; that assignment has OTTER seek as many proofs as possible within the constraints of time and memory.) The first of these two proofs is not the same as the 17-step proof given earlier. It cannot be the same. Do you see why? Clue: What action was taken regarding max_distinct_vars, and what relation did it have to the richness of the given 17-step proof?

The answer rests with the Second Input File and, of course, a glance at the earlier-cited 17-step proof or a review of commentary. The cited proof has variable richness 4, and the assigned value to max_distinct_vars in the Second Input File is 3. Therefore, OTTER is prevented from reproducing the cited 17-step proof—quite nice, yes? Instead, the new 17-step proof has richness 3, and nine of its steps are not among those of the earlier 17-step proof; also of some interest, the new proof has level 9, whereas the earlier has level 7. (This last bit of data, regarding level, I think is the type of data that will, in the long run, provide some clue in the context of the general problem focusing on the nature of proof and of proof search.) Immediately, you might wonder, or ask about, the relation of the second proof, of length 14, to the earlier-given 14-step proof. They are identical; even the order of the deduced steps is preserved.

As for the deduction of thesis 1, five proofs were found, of respective lengths 35, 33, 32, 30, and 28. (If you wish to study in depth the seven proofs, you simply rely on the Second Input File.) The 28-step proof, the following, merits close scrutiny.

A 28-Step Proof of Thesis 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on lemma.mcs.anl.gov,
Thu May 19 18:16:40 2005
----> UNIT CONFLICT at 1589.48 sec ----> 140249 [binary,140248.1,14.1] $ANS(THESIS_1).

Length of proof is 28. Level of proof is 13.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(j(x,y)) | -P(x) | P(y).
4 [] P(i(x,i(y,x))).
5 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
6 [] P(i(x,j(y,x))).
7 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
8 [] P(i(j(i(x,y),y),j(i(y,x),x))).
9 [] P(j(i(x,y),j(x,y))).
```

14 [] -P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).
28 [hyper,1,4,6] P(i(x,i(y,j(z,y)))).
29 [hyper,1,6,4] P(j(x,i(y,i(z,y)))).
32 [hyper,1,6,9] P(j(x,j(i(y,z),j(y,z)))).
39 [hyper,1,5,4] P(i(i(x,y),i(x,x))).
54 [hyper,1,39,6] P(i(x,x)).
57 [hyper,2,9,7] P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
72 [hyper,2,9,54] P(j(x,x)).
78 [hyper,2,9,8] P(j(j(i(x,y),y),j(i(y,x),x))).
123 [hyper,1,5,28] P(i(i(x,y),i(x,j(z,y)))).
144 [hyper,1,6,72] P(j(x,j(y,y))).
166 [hyper,1,7,32] P(j(j(x,i(y,z)),j(x,j(y,z)))).
287 [hyper,2,57,57] P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))).
380 [hyper,1,7,78] P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))).
945 [hyper,2,287,144] P(j(j(x,j(x,y)),j(x,y))).
950 [hyper,2,287,32] P(j(j(i(x,y),j(j(x,y),z)),j(i(x,y),z))).
962 [hyper,1,6,945] P(j(x,j(j(y,j(y,z)),j(y,z)))).
1126 [hyper,2,57,962] P(j(j(x,j(y,j(y,z))),j(x,j(y,z)))).
1853 [hyper,2,380,29] P(j(j(i(i(x,y),y),y),i(x,y))).
1878 [hyper,2,166,1853] P(j(j(i(i(x,y),y),y),j(x,y))).
1887 [hyper,1,6,1853] P(j(x,j(j(i(i(y,z),z),z),i(y,z)))).
2933 [hyper,2,1126,1878] P(j(j(i(i(x,j(x,y)),j(x,y)),j(x,y)),j(x,y))).
4011 [hyper,1,7,1887] P(j(j(x,j(i(i(y,z),z),z)),j(x,i(y,z)))).
15207 [hyper,1,6,2933] P(j(x,j(j(i(i(y,j(y,z)),j(y,z)),j(y,z)),j(y,z)))).
21356 [hyper,2,4011,950] P(j(j(i(i(x,y),y),j(j(i(x,y),y),y)),i(x,y))).
139999 [hyper,2,21356,15207] P(i(i(x,j(x,y)),j(x,y))).
140018 [hyper,1,4,139999] P(i(x,i(i(y,j(y,z)),j(y,z)))).
140061 [hyper,1,5,140018] P(i(i(x,i(y,j(y,z))),i(x,j(y,z)))).
140248 [hyper,1,140061,123] P(i(i(x,y),j(x,y))).

On the surface, we appear to be losing ground in our goal of a proof of length less than 27. But several points should be noted. First, fifteen steps of the 27-step proof are not among the 28.

Second, the 27-step proof has variable richness 4 (two of its steps show this), whereas the 28-step proof has richness 3. Both proofs have level 13. The size of the 27-step proof—total symbol count of the deduced steps, ignoring commas and parentheses—is 782, whereas that of the 28-step proof is 878. Is the difference in size caused by the constraint of assigning 3 (for the 28-step proof) rather than 4 to max_distinct_vars? After all, in general, you might naturally expect that nothing comes free, that such constraints, even though appealing in the context of richness, will result in some price.

Third, what about the role of the 17 resonators found in the Second Input File? Two of the seventeen formulas that were used as resonators are present in the 28-step proof, the following.

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

Perhaps the two shared formulas are not that crucial. Well, they are the fourth and fifth steps of the 28-step proof.

More generally, what role as resonators, rather than as specific formulas, did the seventeen play? As it turns out, which somewhat surprised me, only the cited two are found (as resonators) in the 28-step proof. The way to determine this is to take the seventeen in one window and the twenty-eight in another window and change all variables to the variable *x* and do a set-theoretic subtraction. (McCune has provided me with a program that does the subtraction—how convenient!) So, was headway being made? What could one do with the 28-step proof?

The answer rests at least in part with the following input file, that one used for the third experiment.

Third Input File

```
set(hyper_res).
assign(max_weight,23).
%  assign(change_limit_after,400).
%  assign(new_max_weight,16).
clear(print_kept).
% clear(for_sub).
set(ancestor_subsume).
set(back_sub).
% clear(set_sub).
assign(max_mem,600000).
%  assign(max_seconds,2).
% set(control_memory).
% assign(report,900).
assign(pick_given_ratio,4).
assign(max_proofs,-1).
%set(order_history).
%set(input_sos_first).
%  set(sos_queue).
%set(print_level).
set(order_history).
assign(max_distinct_vars,3).
assign(heat,0).
assign(bsub_hint_wt,2).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
weight(i(i(i(i($(1),$(1)),$(1)),$(1)),$(1)),100).
weight(j(j(j(j($(1),$(1)),$(1)),$(1)),$(1)),100).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).             % Modus
-P(j(x,y)) | -P(x) | P(y).             % Modus
%  -P(i(i(A,B),j(A,B))) |
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).    % Lemmas
end_of_list.

list(sos).
%  Axioms
P(i(x,i(y,x))).                        % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).    % (A2)
%  P(i(i(i(x,y),x),x)).                % (A3)
P(i(x,j(y,x))).                        % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).    % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).      % (A6)
%  P(i(j(j(x,y),x),x)).                % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).         % (A8)
P(j(i(x,y),j(x,y))).                   % (A9)
end_of_list.
```

```
list(passive).
-P(i(j(a1,j(a2,a3)),j(a2,j(a1,a3)))) | $ANS(A6).
-P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
-P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
-P(i(a1,j(a2,a1))) | $ANS(a4).
-P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).              % Lemma
-P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).     % Lemma
-P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).     % Lemma
end_of_list.

list(demodulators).
(P(i(i(i(x,y),x),x)) = junk).  %  A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk).  %  A6
(P(i(j(j(x,y),x),x)) = junk).  %  A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

list(hints).
%  Following 14 from temp.spinks1.depax37.out1e prove A3 dependent on 1 2 4 5 8 9.
P(i(x,i(y,i(z,y)))).
P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
P(i(i(x,y),i(x,i(z,y)))).
P(i(i(x,y),i(x,x))).
P(j(i(x,y),i(x,i(z,y)))).
P(i(i(x,i(x,y)),i(x,y))).
P(j(i(x,i(x,y)),i(x,y))).
P(j(i(i(x,y),x),x)).
P(j(x,j(i(i(y,z),y),y))).
P(j(j(x,i(i(y,z),y)),j(x,y))).
P(j(i(i(i(x,y),z),y),i(x,y))).
P(j(i(i(i(i(x,y),x),z),x),x)).
P(j(i(x,i(i(i(x,y),x),z)),i(i(i(x,y),x),z))).
P(i(i(i(x,y),x),x)).
%  Following 28 are a proof of thesis1, with smaller basis, vars=3, quite different from earlier 27,
%  and from the 19 that relied upon a7, taken from temp.spinks2.thesis1.out1e1.
P(i(x,i(y,j(z,y)))).
P(j(x,i(y,i(z,y)))).
P(j(x,j(i(y,z),j(y,z)))).
P(i(i(x,y),i(x,x))).
P(i(x,x)).
P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
P(j(x,x)).
P(j(j(i(x,y),y),j(i(y,x),x))).
P(i(i(x,y),i(x,j(z,y)))).
P(j(x,j(y,y))).
P(j(j(x,i(y,z)),j(x,j(y,z)))).
P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))).
P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))).
P(j(j(x,j(x,y)),j(x,y))).
```

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

list(hot).
-P(i(x,y)) | -P(x) | P(y).          % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

Before I discuss the role of the 28-step proof, you might briefly read through the options and parameters that precede the lists. Such a reading encounters two new items (the following) and reveals the use of a strategy, Veroff's powerful *hints strategy*, to replace the use of the resonance strategy.

```
assign(bsub_hint_wt,2).
set(keep_hint_subsumers).
```

Rather than an in-depth discussion, the following sentence taken from Chapter 3 of the book titled *Automated Reasoning and the Discovery of Missing and Elegant Proofs* provides what is needed here. In contrast to a resonator that treats all variables as indistinguishable and, therefore, focuses on equivalence classes of formulas or equations, a hint treats the variables precisely as written and focuses on items that are identical to the hint (which, of course, includes alphabetic variants), subsume the hint, or are subsumed by the hint, depending on the included options. The first of the two cited items assigns a small value, 2, to any newly deduced item that is related to a hint as given in the preceding sentence. The second item has the program retain any deduced conclusion that subsumes a hint, where hints are placed in list(hints). I chose to switch from resonators to hints in part because OTTER runs much faster with the latter than with the former. The rest of the explanation for switching can best be described as whim, influenced by the results of experiments in various areas of logic.

As for the use of the key result of the second experiment, its 28 deduced steps were placed as hints in list(hints). I also placed in that list as hints correspondents of the 14 deduced steps of the proof that showed *A*3 dependent. I believe the reason I chose the 14 steps over the steps of either 17-step proof rests with the not-fully-recognized view that the first 17-step proof had served its purpose. Perhaps also relevant—and I am simply trying to guess at unconscious or semiconscious thought influenced by years and years of experimentation—the 14-step proof was the shortest path from the hypotheses to conclusion in the object of the first experiment. Temporary failure: This third experiment merely reproduced the 28-step proof.

Therefore, in the fourth experiment, I turned to the use of a methodology I developed some years ago for seeking shorter proofs. The means is a nonstandard use of demodulation, a procedure typically used for simplification and canonicalization. (Weighting can be used also.) The steps of a given proof are blocked, typically but not always, one at a time by demodulating the corresponding formula or equation to junk; see the list(demodulators) in, for example, the Third Input File. (I chose not to discuss that list earlier to avoid having too many balls in the air at once. You find in that list the means I used to block the use of three of the original axioms so that they will not participate even at the deduced level.) The idea is to force the program to avoid completing the proof in hand; after all, one of its steps has been outlawed. If all goes well, at least one of the *n* runs will indicate that a shorter proof exists, where *n* is the length of the proof in focus. If a shorter proof is found, the methodology calls for adding the corresponding demodulator to list(demodulators) and beginning anew; in other words, iteration is the name of the game.

The iterative approach worked, yielding the following 26-step proof.

A 26-Step Proof of Thesis 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on lemma.mcs.anl.gov,
Thu May 19 19:09:43 2005
The command was "otter". The process ID is 28822.
----> UNIT CONFLICT at   1.02 sec ----> 1379 [binary,1378.1,14.1] $ANS(THESIS_1).

Length of proof is 26. Level of proof is 12.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(j(x,y)) | -P(x) | P(y).
4 [] P(i(x,i(y,x))).
5 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
6 [] P(i(x,j(y,x))).
7 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
8 [] P(i(j(i(x,y),y),j(i(y,x),x))).
9 [] P(j(i(x,y),j(x,y))).
14 [] -P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).
94 [hyper,1,4,6] P(i(x,i(y,j(z,y)))).
95 [hyper,1,6,4] P(j(x,i(y,i(z,y)))).
97 [hyper,1,5,94] P(i(i(x,y),i(x,j(z,y)))).
98 [hyper,1,5,4] P(i(i(x,y),i(x,x))).
109 [hyper,1,98,6] P(i(x,x)).
128 [hyper,2,9,109] P(j(x,x)).
133 [hyper,2,9,8] P(j(j(i(x,y),y),j(i(y,x),x))).
134 [hyper,2,9,7] P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
139 [hyper,1,6,9] P(j(x,j(i(y,z),j(y,z)))).
142 [hyper,1,6,128] P(j(x,j(y,y))).
147 [hyper,1,7,133] P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))).
152 [hyper,2,134,134] P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))).
156 [hyper,1,7,139] P(j(j(x,i(y,z)),j(x,j(y,z)))).
166 [hyper,2,147,95] P(j(j(i(i(x,y),y),y),i(x,y))).
170 [hyper,2,152,142] P(j(j(x,j(x,y)),j(x,y))).
185 [hyper,1,6,166] P(j(x,j(j(i(i(y,z),z),z),i(y,z)))).
188 [hyper,1,6,170] P(j(x,j(j(y,j(y,z)),j(y,z)))).
203 [hyper,1,7,185] P(j(j(x,j(i(i(y,z),z),z)),j(x,i(y,z)))).
206 [hyper,2,134,188] P(j(j(x,j(y,j(y,z))),j(x,j(y,z)))).
215 [hyper,2,203,9] P(j(i(i(i(x,y),y),y),i(x,y))).
218 [hyper,2,203,206] P(j(j(i(i(x,j(y,z)),j(y,z)),j(y,j(y,z))),i(x,j(y,z)))).
774 [hyper,2,156,215] P(j(i(i(i(x,y),y),y),j(x,y))).
1343 [hyper,2,218,774] P(i(i(x,j(x,y)),j(x,y))).
1357 [hyper,1,4,1343] P(i(x,i(i(y,j(y,z)),j(y,z)))).
1361 [hyper,1,5,1357] P(i(i(x,i(y,j(y,z))),i(x,j(y,z)))).
1378 [hyper,1,1361,97] P(i(i(x,y),j(x,y))).
```

This 26-step proof avoids fourteen steps of the given 27-step proof, and it avoids three steps of the given

28-step proof. Rather than giving an entire fourth input file for the fourth experiment, I note that this file can be obtained from the third by inserting at the beginning of list(demodulators) the following demodulator.

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

The given 26-step proof produced for me great satisfaction, for I had, roughly a year ago, worked very hard to finally obtain the given 27-step proof. Indeed, none of the techniques I had in hand at the time and no amount of computer time enabled me to find a proof of length less than 27. Let me be clear: I was very pleased with the 27-step proof, as was Spinks; it was far more interesting to each of us in part because the original proof, a longer proof of length 38 found by Spinks in 1999, relied on axioms 1, 2, 4, 5, 6, 7, and 9. That 38-step proof, given in Section 5, is in part longer because it relies on but one clause for condensed detachment, the one for the function *j*, omitting the clause for condensed detachment with respect to the function *i*. (The omission of a clause or of an inference rule, as you would no doubt expect, might cause the program to find longer proofs.) But now, with OTTER's indispensable assistance, I had a 26-step proof. Further, the 27-step proof has variable richness 4, whereas the 26-step proof has richness 3. (Hilbert would have been pleased in view of his twenty-fourth problem, found in his notebooks by R. Thiele, a problem whose focus is on simpler proofs.) Even the size was a tiny bit better, 781 versus 782.

I did pause for a bit, if memory serves, in my seeking a further improvement. However, perhaps because of a variable richness of 3 and a level of 12, I then thought that perhaps, just perhaps, I could find an even shorter proof than length 26. So, a fifth experiment was in order, with the following input file.

Fifth Input File

```
set(hyper_res).
assign(max_weight,23).
assign(change_limit_after,1100).
assign(new_max_weight,15).
clear(print_kept).
% clear(for_sub).
set(ancestor_subsume).
set(back_sub).
% clear(set_sub).
assign(max_mem,600000).
%  assign(max_seconds,2).
% set(control_memory).
% assign(report,900).
%  assign(pick_given_ratio,4).
assign(max_proofs,-1).
%set(order_history).
%set(input_sos_first).
set(sos_queue).
%set(print_level).
set(order_history).
assign(max_distinct_vars,3).
assign(heat,0).
assign(bsub_hint_wt,2).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
weight(i(i(i(i($(1),$(1)),$(1)),$(1)),$(1)),100).
weight(j(j(j(j($(1),$(1)),$(1)),$(1)),$(1)),100).
end_of_list.
```

```
list(usable).
-P(i(x,y)) | -P(x) | P(y).            % Modus
-P(j(x,y)) | -P(x) | P(y).            % Modus
%  -P(i(i(A,B),j(A,B))) |
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).      % Lemmas
end_of_list.

list(sos).
%  Axioms
P(i(x,i(y,x))).                      % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).  % (A2)
%  P(i(i(i(x,y),x),x)).              % (A3)
P(i(x,j(y,x))).                      % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).  % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).    % (A6)
%  P(i(j(j(x,y),x),x)).              % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).       % (A8)
P(j(i(x,y),j(x,y))).                 % (A9)
end_of_list.

list(passive).
%  Following negs 26/12/3 prove thesis1.
-P(i(a1,i(a2,j(a3,a2)))) | $ANS(inter).
-P(j(a1,i(a2,i(a3,a2)))) | $ANS(inter).
-P(i(i(a1,a2),i(a1,j(a3,a2)))) | $ANS(inter).
-P(i(i(a1,a2),i(a1,a1))) | $ANS(inter).
-P(i(a1,a1)) | $ANS(inter).
-P(j(a1,a1)) | $ANS(inter).
-P(j(j(i(a1,a2),a2),j(i(a2,a1),a1))) | $ANS(inter).
-P(j(j(a1,j(a2,a3)),j(j(a1,a2),j(a1,a3)))) | $ANS(inter).
-P(j(a1,j(i(a2,a3),j(a2,a3)))) | $ANS(inter).
-P(j(a1,j(a2,a2))) | $ANS(inter).
-P(j(j(j(i(a1,a2),a2),i(a2,a1)),j(j(i(a1,a2),a2),a1))) | $ANS(inter).
-P(j(j(j(a1,j(a2,a3)),j(a1,a2)),j(j(a1,j(a2,a3)),j(a1,a3)))) | $ANS(inter).
-P(j(j(a1,i(a2,a3)),j(a1,j(a2,a3)))) | $ANS(inter).
-P(j(j(i(i(a1,a2),a2),a2),i(a1,a2))) | $ANS(inter).
-P(j(j(a1,j(a1,a2)),j(a1,a2))) | $ANS(inter).
-P(j(a1,j(j(i(i(a2,a3),a3),a3),i(a2,a3)))) | $ANS(inter).
-P(j(a1,j(j(a2,j(a2,a3)),j(a2,a3)))) | $ANS(inter).
-P(j(j(a1,j(i(i(a2,a3),a3),a3)),j(a1,i(a2,a3)))) | $ANS(inter).
-P(j(j(a1,j(a2,j(a2,a3))),j(a1,j(a2,a3)))) | $ANS(inter).
-P(j(i(i(i(a1,a2),a2),a2),i(a1,a2))) | $ANS(inter).
-P(j(j(i(i(a1,j(a2,a3)),j(a2,a3)),j(a2,j(a2,a3))),i(a1,j(a2,a3)))) | $ANS(inter).
-P(j(i(i(i(a1,a2),a2),a2),j(a1,a2))) | $ANS(inter).
-P(i(i(a1,j(a1,a2)),j(a1,a2))) | $ANS(inter).
-P(i(a1,i(i(a2,j(a2,a3)),j(a2,a3)))) | $ANS(inter).
-P(i(i(a1,i(a2,j(a2,a3))),i(a1,j(a2,a3)))) | $ANS(inter).
-P(i(i(a1,a2),j(a1,a2))) | $ANS(inter).
-P(i(j(a1,j(a2,a3)),j(a2,j(a1,a3)))) | $ANS(A6).
-P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
-P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
-P(i(a1,j(a2,a1))) | $ANS(a4).
-P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).                % Lemma
```

```
-P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).      % Lemma
-P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).      % Lemma
end_of_list.

list(demodulators).
(P(i(i(i(x,y),x),x)) = junk).  %  A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk).  %  A6
(P(i(j(j(x,y),x),x)) = junk).  %  A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

list(hints).
%  Following 26/12/3 from temp.spinks2.thesis1.out1e3 prove thesis 1.
P(i(x,i(y,j(z,y)))).
P(j(x,i(y,i(z,y)))).
P(i(i(x,y),i(x,j(z,y)))).
P(i(i(x,y),i(x,x))).
P(i(x,x)).
P(j(x,x)).
P(j(j(i(x,y),y),j(i(y,x),x))).
P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
P(j(x,j(i(y,z),j(y,z)))).
P(j(x,j(y,y))).
P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))).
P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))).
P(j(j(x,i(y,z)),j(x,j(y,z)))).
P(j(j(i(i(x,y),y),y),i(x,y))).
P(j(j(x,j(x,y)),j(x,y))).
P(j(x,j(j(i(i(y,z),z),z),i(y,z)))).
P(j(x,j(j(y,j(y,z)),j(y,z)))).
P(j(j(x,j(i(i(y,z),z),z)),j(x,i(y,z)))).
P(j(j(x,j(y,j(y,z))),j(x,j(y,z)))).
P(j(i(i(i(x,y),y),y),i(x,y))).
P(j(j(i(i(x,j(y,z)),j(y,z)),j(y,j(y,z))),i(x,j(y,z)))).
P(j(i(i(i(x,y),y),y),j(x,y))).
P(i(i(x,j(x,y)),j(x,y))).
P(i(x,i(i(y,j(y,z)),j(y,z)))).
P(i(i(x,i(y,j(y,z))),i(x,j(y,z)))).
P(i(i(x,y),j(x,y))).
end_of_list.

list(hot).
-P(i(x,y)) | -P(x) | P(y).              % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

In view of my earlier commentary, you might wonder about my going against the grain, seeking a proof of length less than 26 when said proof has level 12. After all, I have said that a level-saturation search is ordinarily impractical. Well, as I discuss the Fifth Input File, you will see how I attempted to circumvent the possible drowning as the levels are examined.

By assigning the value 3 to max_distinct_vars, my many experiments say that the size of levels will grow less rapidly by quite a bit than were I to assign a greater value. The assignment of the value 23 to max_weight was based on a quick glance at the 26-step proof, noting that, probably, a greater value was not needed. With that value, however, the search through level 12 would almost certainly take far, far too long. Therefore, I added in the input the instructions to reduce the max_weight to 15 after 1100 clauses had been chosen to initiate inference-rule application. I guessed that a smaller value than 1100 would prevent the program from introducing new formulas into the potentially shorter proof; a larger value might drown the program. The new max_weight of 15 was chosen because I thought that room must be provided, in the context of complexity, for the so-called new formulas. The "set" commands were of the type used earlier, including that relevant to the use of Veroff's hints strategy.

As for lists, two weight templates, the following, were included to slow the growth as the levels were examined, by purging certain sequences of nested functions.

```
weight(i(i(i(i($(1),$(1)),$(1)),$(1)),$(1)),100).
weight(j(j(j(j($(1),$(1)),$(1)),$(1)),$(1)),100).
```

In list(passive), I simply retained various targets in part to monitor progress, of course with the intent that the negation of thesis 1 was the key. The contents of the demodulator list were present to prevent axioms 3, 6, and 7 from participating, even at the deduced level, for I was intent upon continuing the study with the 6-element basis throughout. (In the study reported here, the focus was on 1, 2, 4, 5, 8, and 9, a weaker logic in that *A*7 is indeed independent.) Among the lists, for this experiment, the hints list played a key role. By placing the 26 deduced formulas of the 26-step proof in that list, because of the assignment of 2 to bsub_hint_wt, various deduced formulas would be given a weight of 2. The result would be the retention of some formulas that might otherwise be purged because their weight (complexity, measured in symbol count) was too high.

This fifth experiment did in fact manage to plow through levels 1-12, completing a proof of thesis 1. However, the proof has length 31, which meant that failure was the correct evaluation for the fifth experiment.

Therefore a sixth experiment was conducted, almost identical to the fifth. Perhaps, I conjectured, OTTER had changed the max_weight to 15 before some needed formula was retained, needed if a shorter proof was to be found. In other words, perhaps a key formula of weight strictly greater than 15, if retained, would be used to complete a proof of length less than 26. The only change in the sixth experiment was assigning the value 19 to the new_max_weight, rather than assigning the value 15. The conjecture was that more room was needed, in the context of complexity.

The story has a charming ending: OTTER—even with the use of level saturation—won, yielding the following proof.

A 24-Step Proof of Thesis 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Sun May  8 17:07:27 2005
The command was "otter".  The process ID is 3642.

The following has proofs of lengths  28 27 25 24.
----> UNIT CONFLICT at 951.00 sec ----> 29757 [binary,29756.1,48.1] $ANS(THESIS_1).

Length of proof is 24.  Level of proof is 11.

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

9 [] -P(i(x,y)) |  -P(x) | P(y).
```

10 [] -P(j(x,y)) | -P(x) | P(y).
12 [] P(i(x,i(y,x))).
13 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
14 [] P(i(x,j(y,x))).
15 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
16 [] P(i(j(i(x,y),y),j(i(y,x),x))).
17 [] P(j(i(x,y),j(x,y))).
48 [] -P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).
103 [hyper,9,12,14] P(i(x,i(y,j(z,y)))).
105 [hyper,9,14,12] P(j(x,i(y,i(z,y)))).
109 [hyper,10,17,16] P(j(j(i(x,y),y),j(i(y,x),x))).
111 [hyper,10,17,15] P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
113 [hyper,10,17,14] P(j(x,j(y,x))).
117 [hyper,9,14,17] P(j(x,j(i(y,z),j(y,z)))).
134 [hyper,9,13,103] P(i(i(x,y),i(x,j(z,y)))).
140 [hyper,9,15,109] P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))).
144 [hyper,10,111,111] P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))).
148 [hyper,9,15,113] P(j(j(x,y),j(x,x))).
163 [hyper,9,15,117] P(j(j(x,i(y,z)),j(x,j(y,z)))).
228 [hyper,10,140,105] P(j(j(i(i(x,y),y),y),i(x,y))).
238 [hyper,10,144,117] P(j(j(i(x,y),j(j(x,y),z)),j(i(x,y),z))).
245 [hyper,10,144,148] P(j(j(x,j(x,y)),j(x,y))).
420 [hyper,10,163,228] P(j(j(i(i(x,y),y),y),j(x,y))).
479 [hyper,10,113,245] P(j(x,j(j(y,j(y,z)),j(y,z)))).
976 [hyper,9,14,420] P(j(x,j(j(i(i(y,z),z),z),j(y,z)))).
1160 [hyper,10,111,479] P(j(j(x,j(y,j(y,z))),j(x,j(y,z)))).
2744 [hyper,10,238,976] P(j(i(i(i(x,y),y),y),j(x,y))).
5987 [hyper,10,1160,2744] P(j(i(i(i(x,j(x,y)),j(x,y)),j(x,y)),j(x,y))).
9328 [hyper,10,228,5987] P(i(i(x,j(x,y)),j(x,y))).
14429 [hyper,9,12,9328] P(i(x,i(i(y,j(y,z)),j(y,z)))).
21281 [hyper,9,13,14429] P(i(i(x,i(y,j(y,z))),i(x,j(y,z)))).
29756 [hyper,9,21281,134] P(i(i(x,y),j(x,y))).

The discovery of that 24-step proof, with variable richness 3, startled me and, when brought to the attention of Spinks, elicited the word “fantastic”.

A bit of analysis is in order concerning the fifth and sixth experiments. A set-theoretic subtraction of the deduced steps of the 24-step proof from the set of given clauses in the fifth experiment showed that exactly one clause, the following, did not occur in the output from the fifth experiment.

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

In the output of the sixth experiment, that clause is the 3247th chosen as given clause, with weight 18. That clause is the twentieth deduced step among the twenty-four. Further, the nineteenth is among the given clauses in the output of the fifth experiment, the 1347th. It seems most likely that this weight-18 clause, clause (4802) in the sixth experiment, was deduced after max_weight was changed to 15 in the fifth experiment. Its weight, 18, is determined by symbol count in that it does not match any of the hints in the input file for either the fifth or the sixth experiment. Therefore, the clause was too complex to be retained, having a weight strictly greater than 15. Yes, good fortune was indeed present, especially in view of the fact that the new_max_weight in the sixth experiment was 19—just large enough. Summarizing, an assignment of 15 to new_max_weight occurred before the key clause was deduced, causing OTTER to discard it, whereas it was retained with the assignment of 19.

5. Varied Topics and Some Fine Detail

At this point, I touch on various topics, some of which may be presented in a rather sketchy fashion. The connections between these topics may be mostly in my mind and, indeed, not clear to various readers. Nevertheless, some of these notes may eventually spark ideas for some researcher.

5.1. Spinks' 38-Step Proof

If you would enjoy a comparison of historical significance, here is the first proof, found by Spinks in 1999, that deduces thesis 1 and that avoids the use of condensed detachment with respect to the function *i*. His proof relies on a set of hypotheses not restricted to 1, 2 4, 5, 8, and 9. Also, its variable richness is 4, and its length is 38. Therefore, from the viewpoint of Hilbert's twenty-fourth problem, the 24-step proof (found in Section 4) is indeed satisfying and simpler in a number of ways.

A 38-Step Proof of Thesis 1

```
----> UNIT CONFLICT at 2360.67 sec ----> 21364 [binary,21363.1,10.1] $F.

Length of proof is 38.  Level of proof is 15.

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

1 [] -P(j(x,y)) | -P(x) | P(y).
2 [] P(i(x,i(y,x))).
3 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
5 [] P(i(x,j(y,x))).
6 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
7 [] P(i(j(x,j(y,z)),j(y,j(x,z)))).
8 [] P(i(j(j(x,y),x),x)).
9 [] P(j(i(x,y),j(x,y))).
10 [] -P(i(i(A,B),j(A,B))).
11 [hyper,9,1,8] P(j(j(j(x,y),x),x)).
12 [hyper,9,1,5] P(j(x,j(y,x))).
14 [hyper,9,1,3] P(j(i(x,i(y,z)),i(i(x,y),i(x,z)))).
15 [hyper,9,1,2] P(j(x,i(y,x))).
24 [hyper,15,1,12] P(j(x,j(y,i(z,y)))).
28 [hyper,15,1,5] P(i(x,i(y,j(z,y)))).
31 [hyper,15,1,2] P(i(x,i(y,i(z,y)))).
34 [hyper,6,1,9] P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
36 [hyper,11,1,12] P(j(x,j(j(j(y,z),y),y))).
41 [hyper,7,1,9] P(j(j(x,j(y,z)),j(y,j(x,z)))).
53 [hyper,14,1,2] P(i(i(x,y),i(x,x))).
63 [hyper,28,1,14] P(i(i(x,y),i(x,j(z,y)))).
71 [hyper,53,1,9] P(j(i(x,y),i(x,x))).
77 [hyper,71,1,31] P(i(x,x)).
81 [hyper,77,1,9] P(j(x,x)).
155 [hyper,34,1,36] P(j(j(x,j(j(y,z),y)),j(x,y))).
156 [hyper,34,1,24] P(j(j(x,y),j(x,i(z,y)))).
180 [hyper,63,1,14] P(i(i(i(x,y),x),i(i(x,y),j(z,y)))).
191 [hyper,41,1,81] P(j(x,j(j(x,y),y))).
200 [hyper,41,1,9] P(j(x,j(i(x,y),y))).
204 [hyper,191,1,12] P(j(x,j(y,j(j(y,z),z)))).
242 [hyper,191,1,34] P(j(j(j(x,j(y,z)),j(j(x,y),j(x,z))),u),u)).
261 [hyper,191,1,15] P(j(j(j(x,i(y,x)),z),z)).
```

264 [hyper,191,1,12] P(j(j(j(x,j(y,x)),z),z)).
380 [hyper,200,1,191] P(j(j(j(x,j(i(x,y),y)),z),z)).
965 [hyper,204,1,34] P(j(j(x,y),j(x,j(j(y,z),z)))).
5836 [hyper,155,1,156] P(j(j(j(i(x,y),z),y),i(x,y))).
13899 [hyper,965,1,264] P(j(x,j(j(j(y,x),z),z))).
13901 [hyper,965,1,261] P(j(x,j(j(i(y,x),z),z))).
13983 [hyper,13899,1,41] P(j(j(j(x,y),z),j(y,z))).
13999 [hyper,13901,1,41] P(j(j(i(x,y),z),j(y,z))).
14010 [hyper,13983,1,242] P(j(j(x,y),j(j(z,x),j(z,y)))).
14041 [hyper,13999,1,380] P(j(x,j(i(i(y,x),z),z))).
14247 [hyper,14041,1,41] P(j(i(i(x,y),z),j(y,z))).
14447 [hyper,14247,1,180] P(j(x,i(i(x,y),j(z,y)))).
16121 [hyper,14010,1,41] P(j(j(x,y),j(j(y,z),j(x,z)))).
17514 [hyper,16121,1,14447] P(j(j(i(i(x,y),j(z,y)),u),j(x,u))).
21363 [hyper,17514,1,5836] P(i(i(x,y),j(x,y))).

5.2. Subproofs

Relevant to my experiments spread over time, a comparison of the 24-step proof with the 27-step proof (both given in Section 4) shows that sixteen of its steps are not among the twenty-seven. That observation naturally—at least for, me because of some unexpected finds in the past few years—leads to a brief discussion of *proper subproofs,* in the set-theoretic sense, in other words, where specific parentage is ignored and just the deduced formulas are considered. For that discussion, I have in hand three proofs of lengths 43, 43, and 44. The first two proofs are identical, and each is a proper subproof of the third in the set-theoretic sense. Stated slightly differently, the third proof has one formula, the following, not in the first or (identical) second.

67 [hyper,9,12,14] P(i(x,i(y,j(z,y)))).

This extra clause is the first deduced clause in the third proof, that of length 44. It is used, with an input clause, to obtain one deduced clause and only one; here are the two clauses, the input clause and the child.

13 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
92 [hyper,9,13,67] P(i(i(x,y),i(x,j(z,y)))).

A brief review of the last few sentences might lead you to the conjecture that clause (92) is in the first two proofs; but, since clause (67) is not (even with a different number), the correspondent to clause (92) in the first two proofs must have different parents. You are indeed correct, and here are the three clauses of interest.

15 [] P(i(x,j(y,x))).
119 [hyper,10,100,13] P(i(i(x,y),i(i(z,x),i(z,y)))).
139 [hyper,10,119,15] P(i(i(x,y),i(x,j(z,y)))).

Sometimes in the process of refining a proof with respect to length, one encounters a sequence of proofs, each one step shorter than its predecessor, and—so piquant—each a proper subproof of its predecessor, in the just-illustrated set-theoretic sense. In such an event, typically the corresponding input files are almost identical to each other, except that the *n*th (with *n* strictly greater than 1) has one added demodulator when compared with its predecessor. The demodulators correspond to blocking a step of the proof yielded by the predecessor.

As for the three proofs discussed, of respective lengths 43, 43, and 44, their input files were almost alike. In the first, the following demodulator was adjoined, to block the retention of the enclosed formula.

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

Weighting could have been used in place of demodulation by including a resonator corresponding to the enclosed formula with an assignment of the value *k*, where *k* is strictly greater than the value assigned to max_weight. When weighting was used, to my surprise, OTTER returned the cited 44-step proof, that with clause (67) as its first step. To avoid retention of this formula by means of weighting, rather than by using demodulation, I had to include a resonator corresponding to clause (67), assigning a value strictly greater than that assigned to max_weight. If you would like a detailed analysis of why such is the case, I cannot, at this time, give such. I can only observe that weighting focuses on the functional shape, on elements of the equivalence class (in this case) of the included resonator. In contrast, demodulation focuses on instances of the demodulator. When the program deduces clauses in a different order, which can happen when weighting is used rather than demodulation (for proof shortening), the consequences are hard to predict. Yes, the various procedures are indeed interconnected and complex.

Still with proper subproofs as the focus, I found the following two proofs quite intriguing. They are intriguing in part because the second is a proper subproof of the first, if the history of each deduction is ignored; indeed, with the focus on deduced steps, all of the formulas of the second proof are among those of the first. The two proofs are also intriguing because they illustrate yet again that a child (deduced formula) can have different sets of parents—quite a contrast to the animal world in which an offspring (ordinarily) has unique parents.

The goal of the study under discussion was to prove *A*7 dependent in an extension of the *BCSK* logic. The extension is obtained by adjoining six clauses, three focusing on the function *o* for logical **or** and three on *a* for logical **and**. Here are the relevant clauses; the negative clause is the one to prove from the positive clauses with the two forms of condensed detachment featured in this notebook.

```
P(i(x,i(y,x))).                        % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).    % (A2)
P(i(i(i(x,y),x),x)).                   % (A3)
P(i(x,j(y,x))).                        % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).    % (A5)
P(i(j(x,j(y,z)),j(y,j(x,z)))).         % (A6)
P(i(j(j(x,y),x),x)).                   % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).         % (A8)
P(j(i(x,y),j(x,y))).                   % (A9)
%
P(j(x,o(x,y))).                        % (A11)
P(i(y,o(x,y))).                        % (A12)
P(j(j(x,z),j(j(y,z),j(o(x,y),z)))).    % (A13)
P(i(a(x,y),x)).                        % (A14)
P(j(a(x,y),y)).                        % (A15)
P(i(i(x,y),i(i(x,z),i(x,a(y,z))))).    % (A16)
-P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
```

I have separated the clauses through *A*9 from the next six to mark the extension of the *BCSK* logic. As you no doubt have guessed, in my study I commented out both *A*3 and *A*6, as is consistent with the material preceding this section. I also commented out *A*7, the target to be proved dependent. Just as a reminder, *A*7 is not dependent on the set consisting of 1, 2, 4, 5, 8, and 9; 3 and 6 are.

The discovery of this dependency (within the extension of the *BCSK* logic) regarding *A*7 was most unexpected both for me and for Spinks. How exciting, to discover something that perhaps nobody knew before! For one aspect, as noted, *A*7 is independent among the original nine axioms used to study this logic. After many experiments, I had found a 35-step proof establishing the dependency, a proof I give shortly. That proof relies on 1, 2, 4, 5, 8, and 9 and three axioms in the function *o* for logical **or**. Very, very recently, I had a notion concerning finding an even shorter proof. The idea is this: Reintroduce the axioms that were being avoided, 3 and 6, and see whether OTTER returns a proof of length less than 35. More generally, one could adjoin dependent axioms or lemmas to get a shorter proof, then take that proof and refine it, then remove the adjoined items in search of a breakthrough.

Proof 1 of the Dependence of A7

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on theorem.mcs.anl.gov,
Sun Mar 20 12:31:56 2005
The command was "otter". The process ID is 20352.
----> UNIT CONFLICT at 0.09 sec ----> 904 [binary,903.1,24.1] $ANS(a7).

Length of proof is 35. Level of proof is 20.

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

10 [] -P(i(x,y)) | -P(x) | P(y).
11 [] -P(j(x,y)) | -P(x) | P(y).
12 [] P(i(x,i(y,x))).
13 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
14 [] P(i(x,j(y,x))).
15 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
16 [] P(i(j(i(x,y),y),j(i(y,x),x))).
17 [] P(j(i(x,y),j(x,y))).
18 [] P(j(x,o(x,y))).
19 [] P(i(y,o(x,y))).
20 [] P(j(j(x,z),j(j(y,z),j(o(x,y),z)))).
24 [] -P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
130 [hyper,10,13,13] P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
133 [hyper,10,12,14] P(i(x,i(y,j(z,y)))).
135 [hyper,10,12,15] P(i(x,i(j(y,j(z,u)),j(j(y,z),j(y,u))))).
138 [hyper,11,17,16] P(j(j(i(x,y),y),j(i(y,x),x))).
139 [hyper,11,17,15] P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))).
140 [hyper,11,17,14] P(j(x,j(y,x))).
142 [hyper,11,17,12] P(j(x,i(y,x))).
180 [hyper,10,130,133] P(i(i(x,i(j(y,x),z)),i(x,z))).
197 [hyper,11,140,140] P(j(x,j(y,j(z,y)))).
244 [hyper,10,180,135] P(i(j(x,y),j(j(z,x),j(z,y)))).
285 [hyper,11,17,244] P(j(j(x,y),j(j(z,x),j(z,y)))).
290 [hyper,10,244,142] P(j(j(x,y),j(x,i(z,y)))).
298 [hyper,10,15,285] P(j(j(j(x,y),j(z,x)),j(j(x,y),j(z,y)))).
347 [hyper,11,298,197] P(j(j(j(x,y),z),j(y,z))).
362 [hyper,11,285,347] P(j(j(x,j(j(y,z),u)),j(x,j(z,u)))).
qq371 [hyper,11,347,138] P(j(x,j(i(x,y),y))).
qq416 [hyper,11,362,139] P(j(j(x,j(y,z)),j(y,j(x,z)))).
449 [hyper,11,285,371] P(j(j(x,y),j(x,j(i(y,z),z)))).
488 [hyper,11,416,416] P(j(x,j(j(y,j(x,z)),j(y,z)))).
qq506 [hyper,11,416,138] P(j(i(x,y),j(j(i(y,x),x),y))).
559 [hyper,11,449,18] P(j(x,j(i(o(x,y),z),z))).
584 [hyper,11,139,488] P(j(j(x,j(y,j(x,z))),j(x,j(y,z)))).
603 [hyper,11,506,19] P(j(j(i(o(x,y),y),y),o(x,y))).
604 [hyper,11,506,14] P(j(j(i(j(x,y),y),y),j(x,y))).
657 [hyper,11,584,559] P(j(x,j(i(o(x,y),j(x,z)),z))).
681 [hyper,11,416,657] P(j(i(o(x,y),j(x,z)),j(x,z))).
698 [hyper,11,603,681] P(o(x,j(x,y))).
709 [hyper,11,488,698] P(j(j(x,j(o(y,j(y,z)),u)),j(x,u))).
727 [hyper,11,709,140] P(j(x,x)).
738 [hyper,11,20,727] P(j(j(x,y),j(o(y,x),y))).

767 [hyper,11,709,738] P(j(j(j(x,y),x),x)).
814 [hyper,11,285,767] P(j(j(x,j(j(y,z),y)),j(x,y))).
845 [hyper,11,814,604] P(j(j(i(j(j(x,y),x),x),x),x)).
851 [hyper,11,814,290] P(j(j(j(i(x,y),z),y),i(x,y))).
903 [hyper,11,851,845] P(i(j(j(x,y),x),x)).

The significance of placing “qq” to mark three of the deduced formulas will become clear shortly; they are significant children, but they have different parents in the second proof.

Proof 2 of the Dependence of A7

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on lemma.mcs.anl.gov,
Sun Jun 19 19:27:20 2005
The command was "otter".  The process ID is 21259.
----> UNIT CONFLICT at   0.08 sec ----> 879 [binary,878.1,25.1] $ANS(a7).

Length of proof is 29.  Level of proof is 15.

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

10 [] -P(i(x,y)) | -P(x) | P(y).
11 [] -P(j(x,y)) | -P(x) | P(y).
12 [] P(i(x,i(y,x))).
13 [] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
14 [] P(i(x,j(y,x))).
15 [] P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).
16 [] P(i(j(x,j(y,z)),j(y,j(x,z)))).
17 [] P(i(j(i(x,y),y),j(i(y,x),x))).
18 [] P(j(i(x,y),j(x,y))).
19 [] P(j(x,o(x,y))).
20 [] P(i(y,o(x,y))).
21 [] P(j(j(x,z),j(j(y,z),j(o(x,y),z)))).
25 [] -P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
96 [hyper,10,13,13] P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
100 [hyper,10,12,14] P(i(x,i(y,j(z,y)))).
104 [hyper,10,12,15] P(i(x,i(j(y,j(z,u)),j(j(y,z),j(y,u))))).
109 [hyper,11,18,17] P(j(j(i(x,y),y),j(i(y,x),x))).
rr110 [hyper,11,18,16] P(j(j(x,j(y,z)),j(y,j(x,z)))).
112 [hyper,11,18,14] P(j(x,j(y,x))).
114 [hyper,11,18,12] P(j(x,i(y,x))).
rr115 [hyper,10,16,18] P(j(x,j(i(x,y),y))).
189 [hyper,10,96,100] P(i(i(x,i(j(y,x),z)),i(x,z))).
rr204 [hyper,10,16,109] P(j(i(x,y),j(j(i(y,x),x),y))).
208 [hyper,11,110,110] P(j(x,j(j(y,j(x,z)),j(y,z)))).
290 [hyper,10,189,104] P(i(j(x,y),j(j(z,x),j(z,y)))).
299 [hyper,11,204,20] P(j(j(i(o(x,y),y),y),o(x,y))).
300 [hyper,11,204,14] P(j(j(i(j(x,y),y),y),j(x,y))).
337 [hyper,10,15,208] P(j(j(x,j(y,j(x,z))),j(x,j(y,z)))).
370 [hyper,10,290,115] P(j(j(x,y),j(x,j(i(y,z),z)))).
371 [hyper,10,290,114] P(j(j(x,y),j(x,i(z,y)))).
466 [hyper,11,370,19] P(j(x,j(i(o(x,y),z),z))).
541 [hyper,11,337,466] P(j(x,j(i(o(x,y),j(x,z)),z))).
625 [hyper,11,110,541] P(j(i(o(x,y),j(x,z)),j(x,z))).
```

692 [hyper,11,299,625] P(o(x,j(x,y))).
709 [hyper,11,208,692] P(j(j(x,j(o(y,j(y,z)),u)),j(x,u))).
738 [hyper,11,709,112] P(j(x,x)).
752 [hyper,11,21,738] P(j(j(x,y),j(o(y,x),y))).
771 [hyper,11,709,752] P(j(j(j(x,y),x),x)).
822 [hyper,10,290,771] P(j(j(x,j(j(y,z),y)),j(x,y))).
845 [hyper,11,822,371] P(j(j(j(i(x,y),z),y),i(x,y))).
846 [hyper,11,822,300] P(j(j(i(j(j(x,y),x),x),x),x)).
878 [hyper,11,845,846] P(i(j(j(x,y),x),x)).

A comparison of the two given proofs might produce food for thought, as it did for me. For example, all twenty-nine of the deduced formulas in the second are among the thirty-five deduced in the first. Next, the three formulas (clauses) marked with qq in the first proof are those three that are marked with rr in the second. Then, a glance at the history of those marked with rr in the second shows that, for each, one of the parents is *A*6. The inclusion of *A*6, though a dependent axiom, in the input enabled OTTER to complete a 29-step proof, in contrast to the 35-step proof. So, in a cursory sense—in this case, set-theoretic—ignoring parentage, the 29-step proof is a proper subproof of the 35-step proof. (Any discussion of short proofs, or of shortest proof, or of shorter proof, necessitates identification of the underlying axiom set, as well as the inference rule or rules being used; the more axioms present, the greater the likelihood of finding a short proof. Indeed, to be absurd, if one includes enough in the input, at the axiomatic level, then one can always find, for a given theorem, a proof of length 1.)

The former, Proof 1, is the shortest proof I have found in the given context, which offers a challenge if you are interested. Specifically, does there exist a proof of length strictly less than 35 (applications of condensed detachment for the two functions *i* and *j*) showing *A*7 dependent on 1, 2, 4, 5, 8, 9, and the three formulas involving *o*? The removal of *A*6 at both the axiomatic and deduced levels does not prevent a proof from being completed, that of length 35, for example.

The observations made in the context of the two proofs might provide yet one more clue concerning proof finding, proof shortening, and proof search. You see that new parents were found for each of the three children, of *A*6, in the second proof, new parents used to complete the first proof. Therefore, is there merit in the context of proof shortening for an approach that starts with finding a proof relying on dependent axioms (or lemmas), refining it with respect to length, and then turning to the real goal of finding a short proof that avoids dependent axioms at the axiomatic and deduced levels?

At this point, you might indeed have concluded that far too much emphasis is being placed on proof shortening. After all, for many in logic, and mathematics, *any* proof suffices; and, for some, just the knowledge that the theorem is true is enough. Therefore, perhaps the real emphasis should be on finding a so-called first proof. Well, it seems to me that what I offer here, as well as what I intend to offer in future notebooks, can be applied to finding a first proof, to proving some purported theorem. At least minimally, I hope, some reader of one of the planned essays will see how to adapt the approaches to proving theorems when no proof is as yet in hand. In other words, I am implicitly offering a topic for research at the doctoral level or more. If you are skeptical, which is most understandable, I now briefly tell the story of such an occurrence, of finding a proof with similar methodology for a theorem announced but not proved.

The relevant story concerns a 23-letter single axiom, the following, for classical propositional calculus, an axiom provided by Lukasiewicz.

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

In his 1930s paper, he offered this axiom, but he did not include a proof that the formula suffices as a complete axiomatization. For years I had been studying proof refinement, mostly in the context of proof shortening, unaware for most of that time of Hilbert's twenty-fourth problem. A spin-off of that study was my attempt to find the first fully automated proof for a 21-letter single axiom, the following, for that area of logic (propositional logic), an axiom supplied years after Lukasiewicz by C. A. Meredith.

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

When I succeeded in finding that fully automated proof, different (as expected) from Meredith's, I applied the methodology to seeking a proof for the 23-letter axiom, the first proof (from what I know) ever completed. the methodology triumphed: I succeeded in finding a proof for the 23-letter formula, or, no doubt, I would not be telling this story; see *Automated Reasoning and the Discovery of Missing and Elegant Proofs* for copious detail about the two successes. (If your mind intersects with mine to a fair extent, which might worry you, you will find two facts from that earlier study interesting. First, my studies eventually led to a 38-step proof showing the Meredith formula sufficient, three steps shorter than the Meredith proof, and he was interested in finding short proofs. Second, I found a 50-step proof showing the Lukasiewicz formula sufficient to serve as a single axiom. OTTER and various strategies and methodologies played the key role. For another open question, I know of no shorter proof, respectively, than the cited 38-step proof and the cited 50-step proof.)

Before turning, in the next subsection, to dispelling a myth, the following delightful example (taken from history focusing on the year 2004) will provide yet more insight into unusual aspects of research. As was the case with the focus on the Lukasiewicz 23-letter single axiom and the Meredith 21-letter single axiom, the area of logic of concern is again classical propositional calculus. However—in sharp contrast to my study of the two cited single axioms and in contrast to the heavy emphasis on the *BCSK* logic—in the problem you are about to meet, the inference rule condensed detachment is *not* a valid rule for drawing conclusions. For this problem, the credit goes to H. Hiz and his presentation in 1959. In place of condensed detachment, Hiz offered three inference rules to use, the following, which are captured with three clauses and the use of hyperresolution.

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

The function *n* denotes negation. Hiz offered the following 2-basis.

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

At the most general level—and not that which will be the focus here—one problem asks you to show, as Hiz did, that the use of the three given inference rules suffices to show that the cited 2-basis provides an axiomatization for classical propositional calculus. The problem here is, instead, to produce what was missing for more than four decades (if I have been correctly informed), namely, a first-order proof that the given 2-basis offers what is needed. In other words, a proof existed, but not the type of proof featured in much of automated reasoning. So, the question you might ponder asks about how to proceed to find a proof that is in the spirit of the proofs offered, say, in this notebook. To give you a small amount of time to consider what you might do, I supply just a taste of some background.

This question, of finding the so-called missing proof that is indeed a first-order proof, was brought to my attention (if memory serves) in May 2004 by B. Fitelson. Three years earlier, Fitelson, Z. Ernst, K. Harris, and I had successfully collaborated on the use of OTTER to find proofs in, among other areas, *C*4 and *C*5. Immediately I was intrigued. After all, I almost always prefer a first-order proof to any other type. Well, time is up: I shall now finish the story by telling you how I proceeded, giving you an appropriate input file.

I proceeded as I so often do: I emulated what had worked before in moderately similar situations. The choice of targets was clear; instruct the program to attempt to complete a proof of one or more known bases for propositional calculus. Therefore, in list(usable) of the input file I used, I placed the negation of various bases. The key question focused on strategy, strategy to restrict the reasoning and strategy to direct it. I knew that iteration might be required. Although I would have preferred to block the retention of any deduced conclusion that contained a term of the form *n*(*n*(*t*)) for some term *t*, I thought that restriction might be an unwise choice. Such terms are called double-negation terms. So, instead, I blocked (as you will see in the input file I shall give) the use of triple negation. As for a direction strategy, the use of resonators had proved powerful in the past; therefore, I chose to use them. The choice of which resonators to rely on was obvious, at least, in the beginning. I chose two sets of resonators. The first set, given the most preference, consisted of individual members of known axiom systems. The second set consisted of what

Lukasiewicz called theses, theorems that can be proved, sixty-eight of them, given a bit less preference. I also included many, many hints, hints that came from I know not where at this time; that information is lost to history. The following input file was used in my first attempt.

Original Input File for Studying the Hiz Approach

```
set(hyper_res).
assign(max_weight,20).
% assign(change_limit_after,350).
% assign(new_max_weight,24).
assign(max_proofs,-1).
% assign(max_seconds,30).
% set(process_input).
% set(ancestor_subsume).
% set(back_sub).

clear(print_kept).
clear(print_back_sub).

assign(max_distinct_vars,7).
assign(pick_given_ratio,2).

assign(max_mem,600000).
% assign(report,3600).

set(order_history).
set(input_sos_first).
% assign(heat,0).
% assign(dynamic_heat_weight,1).

assign(bsub_hint_add_wt,-1000).
set(degrade_hints2).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
%  Following are members of known axiom systems for two-valued.
weight(P(i(i(x,y),i(i(y,r),i(x,r)))),1).
weight(P(i(i(n(x),x),x)),1).
weight(P(i(x,i(n(x),y))),1).
weight(P(i(y,i(x,y))),1).
weight(P(i(i(i(x,y),z),i(y,z))),1).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),1).
weight(P(i(i(y,z),i(i(x,y),i(x,z)))),1).
weight(P(i(i(x,i(x,y)),i(x,y))),1).
weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),1).
weight(P(i(i(i(x,y),z),i(n(x),z))),1).
weight(P(i(n(n(x)),x)),1).
weight(P(i(x,n(n(x)))),1).
weight(P(i(i(x,y),i(n(y),n(x)))),1).
weight(P(i(i(n(x),n(y)),i(y,x))),1).
weight(P(i(i(x,y),i(i(n(x),y),y))),1).
weight(P(i(i(n(x),z),i(i(y,z),i(i(x,y),z)))),1).
weight(P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z))))),1).
```

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

```
weight(P(i(i(i(i(x,y),y),z),i(i(n(x),y),z))),2).
weight(P(i(i(n(x),y),i(i(x,z),i(i(z,y),y)))),2).
weight(P(i(i(i(i(x,y),i(i(y,z),z)),u),i(i(n(x),z),u))),2).
weight(P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))),2).
weight(P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))),2).
weight(P(i(i(x,y),i(i(z,y),i(i(n(x),z),y)))),2).
weight(P(i(i(n(n(x)),y),i(x,y))),2).
weight(P(i(x,i(y,y))),2).
weight(P(i(n(i(x,x)),y)),2).
weight(P(i(i(n(x),n(i(y,y))),x)),2).
weight(P(i(n(i(x,y)),x)),2).
weight(P(i(n(i(x,y)),n(y))),2).
weight(P(i(n(i(x,n(y))),y)),2).
weight(P(i(x,i(n(y),n(i(x,y))))),2).
weight(P(i(x,i(y,n(i(x,n(y)))))),2).
weight(P(n(i(i(x,x),n(i(y,y))))),2).
%  Following is Lukasiewicz's 23-letter single axiom.
%  weight(P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))),1).
%  %  Following is recursive tail strategy.
%  weight(i($(1),$(2)),1).
end_of_list.

list(usable).
% Hiz's three rules of inference [Note: detachment is NOT valid!]
-P(i(x,y)) | -P(i(y,z)) | P(i(x,z)).
-P(i(x,i(y,z))) | -P(i(x,y)) | P(i(x,z)).
-P(i(n(x),y)) | -P(i(n(x),n(y))) | P(x).

% several known bases for 2-valued
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) |
-P(i(n(n(p)),p)) | -P(i(p,n(n(p)))) | -P(i(i(p,q),i(n(q),n(p)))) |
-P(i(i(p,i(q,r)),i(q,i(p,r)))) |
$ANS(step_allFrege_18_35_39_40_46_21).  % 21 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) |
-P(i(i(q,r),i(i(p,q),i(p,r)))) | -P(i(p,i(n(p),q))) |
-P(i(i(p,q),i(i(n(p),q),q))) | -P(i(i(p,i(p,q)),i(p,q))) |
$ANS(step_allHilbert_18_21_22_3_54_30).  % 30 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) |
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_allBEH_Church_FL_18_35_49).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) |
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) |
$ANS(step_allLuka_x_19_37_59).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) |
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) |
$ANS(step_allWos_x_19_37_60).
-P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) |
-P(i(p,i(n(p),q))) | $ANS(step_allLuka_1_2_3).
end_of_list.

list(sos).
% Hiz's two axioms
P(i(n(i(x,y)),x)).
P(i(n(i(x,y)),n(y))).
```

```
end_of_list.

list(demodulators).
% demods for purging unwanted formulae
n(n(n(x))) = junk.
% n(n(x)) = junk.
i(x,junk) = junk.
i(junk,x) = junk.
n(junk) = junk.
P(junk) = $T.
end_of_list.

list(passive).
%  Following are members of known axiom systems for two-valued.
-P(i(i(p,q),i(i(q,r),i(p,r)))) | $ANS(step_L1).
-P(i(i(n(p),p),p)) | $ANS(step_L2).
-P(i(p,i(n(p),q))) | $ANS(step_L3).
-P(i(q,i(p,q))) | $ANS(step_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANS(step_19).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(step_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(step_22).
-P(i(i(p,i(p,q)),i(p,q))) | $ANS(step_30).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(step_35).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANS(step_37).
-P(i(n(n(p)),p)) | $ANS(step_39).
-P(i(p,n(n(p)))) | $ANS(step_40).
-P(i(i(p,q),i(n(q),n(p)))) | $ANS(step_46).
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_49).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANS(step_54).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANS(step_60).

%  Following are negations of theses 4-71.
-P(i(i(i(i(q,r),i(p,r)),s),i(i(p,q),s))) | $ANS(neg_th_04).
-P(i(i(p,i(q,r)),i(i(s,q),i(p,i(s,r))))) | $ANS(neg_th_05).
-P(i(i(p,q),i(i(i(p,r),s),i(i(q,r),s)))) | $ANS(neg_th_06).
-P(i(i(t,i(i(p,r),s)),i(i(p,q),i(t,i(i(q,r),s))))) | $ANS(neg_th_07).
-P(i(i(q,r),i(i(p,q),i(i(r,s),i(p,s))))) | $ANS(neg_th_08).
-P(i(i(i(n(p),q),r),i(p,r))) | $ANS(neg_th_09).
-P(i(p,i(i(i(n(p),p),p),i(i(q,p),p)))) | $ANS(neg_th_10).
-P(i(i(q,i(i(n(p),p),p)),i(i(n(p),p),p))) | $ANS(neg_th_11).
-P(i(t,i(i(n(p),p),p))) | $ANS(neg_th_12).
-P(i(i(n(p),q),i(t,i(i(q,p),p)))) | $ANS(neg_th_13).
-P(i(i(i(t,i(i(q,p),p)),r),i(i(n(p),q),r))) | $ANS(neg_th_14).
-P(i(i(n(p),q),i(i(q,p),p))) | $ANS(neg_th_15).
-P(i(p,p)) | $ANS(neg_th_16).
-P(i(p,i(i(q,p),p))) | $ANS(neg_th_17).
-P(i(q,i(p,q))) | $ANS(neg_th_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANS(neg_th_19).
-P(i(p,i(i(p,q),q))) | $ANS(neg_th_20).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(neg_th_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(neg_th_22).
-P(i(i(i(q,i(p,r)),s),i(i(p,i(q,r)),s))) | $ANS(neg_th_23).
```

```
-P(i(i(i(p,q),p),p)) | $ANS(neg_th_24).
-P(i(i(i(p,r),s),i(i(p,q),i(i(q,r),s)))) | $ANS(neg_th_25).
-P(i(i(i(p,q),r),i(i(r,p),p))) | $ANS(neg_th_26).
-P(i(i(i(p,q),q),i(i(q,p),p))) | $ANS(neg_th_27).
-P(i(i(i(i(r,p),p),s),i(i(i(p,q),r),s))) | $ANS(neg_th_28).
-P(i(i(i(p,q),r),i(i(p,r),r))) | $ANS(neg_th_29).
-P(i(i(p,i(p,q)),i(p,q))) | $ANS(neg_th_30).
-P(i(i(p,s),i(i(i(p,q),r),i(i(s,r),r)))) | $ANS(neg_th_31).
-P(i(i(i(p,q),r),i(i(p,s),i(i(s,r),r)))) | $ANS(neg_th_32).
-P(i(i(p,s),i(i(s,i(q,i(p,r))),i(q,i(p,r))))) | $ANS(neg_th_33).
-P(i(i(s,i(q,i(p,r))),i(i(p,s),i(q,i(p,r))))) | $ANS(neg_th_34).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(neg_th_35).
-P(i(n(p),i(p,q))) | $ANS(neg_th_36).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANS(neg_th_37).
-P(i(i(p,n(p)),n(p))) | $ANS(neg_th_38).
-P(i(n(n(p)),p)) | $ANS(neg_th_39).
-P(i(p,n(n(p)))) | $ANS(neg_th_40).
-P(i(i(p,q),i(n(n(p)),q))) | $ANS(neg_th_41).
-P(i(i(i(n(n(p)),q),r),i(i(p,q),r))) | $ANS(neg_th_42).
-P(i(i(p,q),i(i(q,n(p)),n(p)))) | $ANS(neg_th_43).
-P(i(i(s,i(q,n(p))),i(i(p,q),i(s,n(p))))) | $ANS(neg_th_44).
-P(i(i(s,i(q,p)),i(i(n(p),q),i(s,p)))) | $ANS(neg_th_45).
-P(i(i(p,q),i(n(q),n(p)))) | $ANS(neg_th_46).
-P(i(i(p,n(q)),i(q,n(p)))) | $ANS(neg_th_47).
-P(i(i(n(p),q),i(n(q),p))) | $ANS(neg_th_48).
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(neg_th_49).
-P(i(i(i(n(q),p),r),i(i(n(p),q),r))) | $ANS(neg_th_50).
-P(i(i(p,i(q,r)),i(p,i(n(r),n(q))))) | $ANS(neg_th_51).
-P(i(i(p,i(q,n(r))),i(p,i(r,n(q))))) | $ANS(neg_th_52).
-P(i(i(n(p),q),i(i(p,q),q))) | $ANS(neg_th_53).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANS(neg_th_54).
-P(i(i(p,q),i(i(p,n(q)),n(p)))) | $ANS(neg_th_55).
-P(i(i(i(i(p,q),q),r),i(i(n(p),q),r))) | $ANS(neg_th_56).
-P(i(i(n(p),r),i(i(p,q),i(i(q,r),r)))) | $ANS(neg_th_57).
-P(i(i(i(i(p,q),i(i(q,r),r)),s),i(i(n(p),r),s))) | $ANS(neg_th_58).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(neg_th_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANS(neg_th_60).
-P(i(i(p,r),i(i(q,r),i(i(n(p),q),r)))) | $ANS(neg_th_61).
-P(i(i(n(n(p)),q),i(p,q))) | $ANS(neg_th_62).
-P(i(q,i(p,p))) | $ANS(neg_th_63).
-P(i(n(i(p,p)),q)) | $ANS(neg_th_64).
-P(i(i(n(q),n(i(p,p))),q)) | $ANS(neg_th_65).
-P(i(n(i(p,q)),p)) | $ANS(neg_th_66).
-P(i(n(i(p,q)),n(q))) | $ANS(neg_th_67).
-P(i(n(i(p,n(q))),q)) | $ANS(neg_th_68).
-P(i(p,i(n(q),n(i(p,q))))) | $ANS(neg_th_69).
-P(i(p,i(q,n(i(p,n(q)))))) | $ANS(neg_th_70).
-P(n(i(i(p,p),n(i(q,q))))) | $ANS(neg_th_71).
end_of_list.

list(hints2).
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
```

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

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

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

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

As it turned out, one experiment sufficed; indeed, iteration was not needed. OTTER found a proof, a first-order proof showing that the Hiz 2-basis, with his three inference rules, axiomatizes classical propositional calculus. The proof presented to me by McCune's program was the following.

The First First-Order Proof Showing the Hiz 3-Axiom System Suffices

```
----- Otter 3.3d, April 2004 -----
The process was started by wos on lemma.mcs.anl.gov,
Mon May  3 15:13:24 2004
The command was "otter".  The process ID is 11514.
```

-----> EMPTY CLAUSE at 2851.17 sec ----> 528791 [hyper,9,528751,340,471] $ANS(step_allLuka_1_2_3).

Length of proof is 48. Level of proof is 13.

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

1 [] -P(i(x,y))| -P(i(y,z))|P(i(x,z)).
2 [] -P(i(x,i(y,z)))| -P(i(x,y))|P(i(x,z)).
3 [] -P(i(n(x),y))| -P(i(n(x),n(y)))|P(x).
9 [] -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).
10 [] P(i(n(i(x,y)),x)).
11 [] P(i(n(i(x,y)),n(y))).
304 [hyper,1,10,10] P(i(n(i(n(i(x,y)),z)),x)).
307 [hyper,3,11,10] P(i(n(n(x)),x)).
310 [hyper,2,10,11] P(i(n(i(i(n(x),y),x)),y)).
311 [hyper,1,11,11] P(i(n(i(x,i(y,z))),n(z))).
312 [hyper,1,10,11] P(i(n(i(n(i(x,y)),z)),n(y))).
313 [hyper,1,11,10] P(i(n(i(x,i(y,z))),y)).
321 [hyper,1,11,307] P(i(n(i(x,n(y))),y)).
327 [hyper,2,304,321] P(i(n(i(n(i(i(x,y),z)),n(x))),y)).
333 [hyper,1,11,321] P(i(n(i(x,i(y,n(z)))),z)).
340 [hyper,3,310,11] P(i(i(n(x),x),x)).
397 [hyper,2,10,311] P(i(n(i(i(n(x),y),i(z,x))),y)).
401 [hyper,1,311,311] P(i(n(i(x,i(y,i(z,i(u,v))))),n(v))).
404 [hyper,1,11,311] P(i(n(i(x,i(y,i(z,u)))),n(u))).
405 [hyper,1,10,311] P(i(n(i(n(i(x,i(y,z))),u)),n(z))).
410 [hyper,1,311,10] P(i(n(i(x,i(y,i(z,u)))),z)).
439 [hyper,1,311,312] P(i(n(i(x,i(y,i(n(i(z,u)),v)))),n(u))).
471 [hyper,3,10,313] P(i(x,i(n(x),y))).
473 [hyper,3,313,312] P(i(n(i(x,y)),i(y,z))).
482 [hyper,2,10,313] P(i(n(i(i(x,y),i(x,z))),y)).
490 [hyper,1,313,313] P(i(n(i(x,i(n(i(y,i(z,u))),v))),z)).
1150 [hyper,3,327,312] P(i(n(i(i(x,y),y)),n(x))).
1163 [hyper,1,311,327] P(i(n(i(x,i(y,i(n(i(i(z,u),v)),n(z))))),u)).
6446 [hyper,3,473,1150] P(i(i(i(x,y),x),x)).
6455 [hyper,3,333,1150] P(i(i(x,i(y,n(x))),i(y,n(x)))).
6458 [hyper,3,313,1150] P(i(i(x,i(x,y)),i(x,y))).
6510 [hyper,1,11,1150] P(i(n(i(x,i(i(y,z),z))),n(y))).
7295 [hyper,1,313,6446] P(i(n(i(x,i(i(i(y,z),y),u))),y)).
11130 [hyper,3,313,397] P(i(i(n(x),n(y)),i(y,x))).
11323 [hyper,1,6455,11130] P(i(i(x,i(n(y),n(x))),i(x,y))).
12102 [hyper,3,313,401] P(i(x,i(y,i(z,i(u,y))))).
12309 [hyper,1,12102,6458] P(i(x,i(y,i(z,y)))).
13255 [hyper,3,327,405] P(i(n(i(i(x,y),i(z,y))),n(x))).
15274 [hyper,2,313,410] P(i(n(i(x,i(i(y,z),i(y,u)))),z)).
19837 [hyper,3,482,404] P(i(i(x,y),i(x,i(z,y)))).
32456 [hyper,3,490,13255] P(i(i(x,y),i(n(i(z,i(x,u))),y))).
119406 [hyper,3,397,6510] P(i(i(n(x),y),i(i(y,x),x))).
349465 [hyper,3,1163,439] P(i(x,i(y,i(n(i(i(z,u),u)),n(z))))).
349645 [hyper,1,349465,11323] P(i(x,i(y,i(i(y,z),z)))).
386966 [hyper,3,7295,13255] P(i(i(x,y),i(i(i(x,z),x),y))).
431661 [hyper,3,15274,13255] P(i(i(x,i(y,z)),i(i(y,x),i(y,z)))).
432348 [hyper,1,19837,431661] P(i(i(x,y),i(i(z,x),i(z,y)))).

434483 [hyper,2,432348,349645] P(i(i(i(i(x,y),y),z),i(x,z))).
435808 [hyper,1,434483,386966] P(i(i(i(i(x,y),y),z),i(i(i(x,u),x),z))).
440385 [hyper,1,432348,435808] P(i(i(x,y),i(i(i(z,u),z),i(i(z,x),y)))).
442815 [hyper,1,12309,440385] P(i(x,i(i(i(y,z),y),i(i(y,u),i(v,u))))).
443041 [hyper,1,442815,434483] P(i(x,i(y,i(i(y,z),i(u,z))))).
528676 [hyper,1,32456,119406] P(i(i(x,y),i(i(y,i(z,i(x,u))),i(z,i(x,u))))).
528751 [hyper,2,528676,443041] P(i(i(x,y),i(i(y,z),i(x,z)))).

In the run that yielded this proof, OTTER also completed a proof of the Hilbert axiom system, the so-called Church system (actually due to Lukasiewicz), and the Frege system. If, as might be the case, you suspect I loaded the dice, perhaps unconsciously, you can take the given input file and remove all items regarding hints; it works well. Since the use of either input file, that given or that obtained by the cited modification, yields the sought-after first-order proof, I find great satisfaction—especially in that, originally, one experiment sufficed—and view the results as fine evidence for the value of the implied methodology. Yes, that methodology was formulated, I am fairly certain, in my studies aimed at proof shortening.

I close this subsection with one more challenge. I have in hand a 10-step proof that deduces, from the Hiz 2-basis with his three inference rules, the same Lukasiewicz 3-axiom system deduced in the just-given 48-step proof. You might see whether you can find that 10-step proof.

5.3. Dispelling a Myth about Variation in Proof Length

The next topic concerns variation in proof length that you might find in the output of a single run. In Section 4, I cited the negation of the conjunction of two theses to prove, the following.

-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).   % Lemmas

The study focusing on finding a short proof of the join of the two theses, produced (among others) a nice example of how proofs can vary in length within a single run. In particular, a single run with OTTER, with the goal of deducing the (positive form of) two theses, yielded proofs of the following lengths and times.

-----> EMPTY CLAUSE at 286.30 sec ----> 152123 [hyper,11,151064,150326] $ANS(THESIS_23).
Length of proof is 78. Level of proof is 20.
-----> EMPTY CLAUSE at 359.82 sec ----> 191054 [hyper,11,151064,190649] $ANS(THESIS_23).
Length of proof is 74. Level of proof is 20.
-----> EMPTY CLAUSE at 519.17 sec ----> 242565 [hyper,11,242415,190649] $ANS(THESIS_23).
Length of proof is 80. Level of proof is 20.
-----> EMPTY CLAUSE at 2363.69 sec ----> 482743 [hyper,11,482503,190649] $ANS(THESIS_23).
Length of proof is 100. Level of proof is 22.
-----> EMPTY CLAUSE at 3342.04 sec ----> 576364 [hyper,11,575455,575069] $ANS(THESIS_23).
Length of proof is 71. Level of proof is 20.
-----> EMPTY CLAUSE at 12544.81 sec ----> 1143904 [hyper,11,575455,1143204] $ANS(THESIS_23).
Length of proof is 70. Level of proof is 20.
-----> EMPTY CLAUSE at 35519.17 sec ----> 1906693 [hyper,11,575455,1906651] $ANS(THESIS_23).
Length of proof is 70. Level of proof is 21.

A glance at the results just given could easily lead you to ask what is going on. Are the increasing and decreasing lengths caused by poorer and poorer results in the context of proofs of each of theses 2 and 3? Well, here are the results for thesis 2, followed by those for thesis 3.

----> UNIT CONFLICT at 283.41 sec ----> 149622 [binary,149621.1,19.1] $ANS(THESIS_2).
Length of proof is 57. Level of proof is 19.
----> UNIT CONFLICT at 285.21 sec ----> 151065 [binary,151064.1,19.1] $ANS(THESIS_2).
Length of proof is 54. Level of proof is 20.
----> UNIT CONFLICT at 518.53 sec ----> 242416 [binary,242415.1,19.1] $ANS(THESIS_2).
Length of proof is 53. Level of proof is 20.
----> UNIT CONFLICT at 2361.90 sec ----> 482504 [binary,482503.1,19.1] $ANS(THESIS_2).
Length of proof is 50. Level of proof is 22.

----> UNIT CONFLICT at 3335.01 sec ----> 575456 [binary,575455.1,19.1] $ANS(THESIS_2).
Length of proof is 44. Level of proof is 20.

----> UNIT CONFLICT at 283.94 sec ----> 150327 [binary,150326.1,20.1] $ANS(THESIS_3).
Length of proof is 75. Level of proof is 20.
----> UNIT CONFLICT at 359.39 sec ----> 190650 [binary,190649.1,20.1] $ANS(THESIS_3).
Length of proof is 71. Level of proof is 20.
----> UNIT CONFLICT at 3332.11 sec ----> 575070 [binary,575069.1,20.1] $ANS(THESIS_3).
Length of proof is 67. Level of proof is 20.
----> UNIT CONFLICT at 12535.03 sec ----> 1143205 [binary,1143204.1,20.1] $ANS(THESIS_3).
Length of proof is 66. Level of proof is 20.
----> UNIT CONFLICT at 35517.97 sec ----> 1906652 [binary,1906651.1,20.1] $ANS(THESIS_3).
Length of proof is 65. Level of proof is 21.

I leave to you a detailed analysis of the time(s) and length(s) to determine which proofs occurred in which order. Nevertheless, you now have a rather detailed example that can be used to dispel a myth, one asserting that finding shorter and still shorter proofs of a conjunction is facilitated by finding shorter and still shorter proofs of the individual members. What goes wrong? The answer rests with the fact that the appropriate subproofs, of members of a conjunction, can diverge. For example, in the case under discussion, the output file contains proofs of thesis 3 of length 75 and 71 before returning a proof of length 100 for the conjunction of theses 2 and 3. At the same time, the output file returns a sequence of proofs of thesis 2 that begin with one of length 57 and end with one of length 50. The 71-step proof and the 50-step proof, subproofs to be more accurate in the context of the proof of the conjunction, are used in the 100-step proof. In contrast, the first proof of the conjunction of 2 and 3 has length 78, containing a subproof of thesis 2 of length 54 and a subproof of length 75 of thesis 3. By way of further explanation, the 75-step proof contains all but three steps of the 54-step proof, hence a proof of length 78 of the conjunction. In contrast, the 71-step proof contains just twenty-one steps of the 50-step proof, in effect avoiding twenty-nine steps of that proof; hence, because of containing a 71-step subproof, the total proof has length 100. Summarizing, you now have some of the details of a run, whose input file I give almost immediately, showing that shorter subproofs—although signifying progress for the individual members of a conjunction—do not necessarily signal progress for the proof of the entire conjunction.

An Input File Producing a Number of Proofs of Interest

```
set(hyper_res).
assign(max_weight,28).
clear(print_kept).
% clear(for_sub).
set(ancestor_subsume).
set(back_sub).
% clear(set_sub).
assign(max_mem,600000).
%  assign(max_seconds,7).
% set(control_memory).
% assign(report,900).
assign(pick_given_ratio,4).
assign(max_proofs,-1).
%set(order_history).
%set(input_sos_first).
%set(sos_queue).
%set(print_level).
set(order_history).
assign(max_distinct_vars,4).
assign(heat,0).
```

```
%
% Modifications to strategy
%

%
% Clauses
%

list(demodulators).
%  (P(i(i(x,j(y,j(z,u))),i(x,j(j(y,z),j(y,u))))) = junk).
(P(i(i(i(x,y),x),x)) = junk).  %  A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk).  %  A6
(P(i(j(j(x,y),x),x)) = junk).  %  A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

weight_list(pick_and_purge).
%  following 26/12/4 prove thesis1, new as of 05-06-05.
weight(P(i(x,i(y,j(z,y)))),0).
weight(P(j(x,i(y,i(z,y)))),0).
weight(P(i(i(x,y),i(x,j(z,y)))),0).
weight(P(i(i(x,y),i(x,x))),0).
weight(P(i(x,x)),0).
weight(P(j(x,x)),0).
weight(P(j(j(i(x,y),y),j(i(y,x),x))),0).
weight(P(j(j(x,j(y,z)),j(j(x,y),j(x,z)))),0).
weight(P(j(x,j(i(y,z),j(y,z)))),0).
weight(P(j(x,j(y,y))),0).
weight(P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))),0).
weight(P(j(j(j(x,j(y,z)),j(x,y)),j(j(x,j(y,z)),j(x,z)))),0).
weight(P(j(j(x,i(y,z)),j(x,j(y,z)))),0).
weight(P(j(j(i(i(x,y),y),y),i(x,y))),0).
weight(P(j(j(x,j(x,y)),j(x,y))),0).
weight(P(j(x,j(j(i(i(y,z),z),z),i(y,z)))),0).
weight(P(j(x,j(j(y,j(y,z)),j(y,z)))),0).
weight(P(j(j(x,j(i(i(y,z),z),z)),j(x,i(y,z)))),0).
weight(P(j(j(x,j(y,j(y,z))),j(x,j(y,z)))),0).
weight(P(j(i(i(i(x,y),y),y),i(x,y))),0).
weight(P(j(j(i(i(x,j(y,z)),j(y,z)),j(y,j(y,z))),i(x,j(y,z)))),0).
weight(P(j(i(i(i(x,y),y),y),j(x,y))),0).
weight(P(i(i(x,j(x,y)),j(x,y))),0).
weight(P(i(x,i(i(y,j(y,z)),j(y,z)))),0).
weight(P(i(i(x,i(y,j(y,z))),i(x,j(y,z)))),0).
weight(P(i(i(x,y),j(x,y))),0).
%  Following 48/19 prove the theorem, from temp.spinks1.out1w3.
weight(P(i(x,i(y,j(z,y)))),2).
weight(P(j(x,i(y,i(z,y)))),2).
weight(P(j(x,j(y,x))),2).
weight(P(j(x,i(y,x))),2).
```

```
weight(P(j(i(x,i(y,z)),i(i(x,y),i(x,z)))),2).
weight(P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))),2).
weight(P(i(x,i(i(y,i(z,u)),i(i(y,z),i(y,u))))),2).
weight(P(i(i(x,y),i(x,j(z,y)))),2).
weight(P(i(i(x,y),i(x,x))),2).
weight(P(i(i(x,i(y,i(z,u))),i(x,i(i(y,z),i(y,u))))),2).
weight(P(j(i(x,y),i(x,j(z,y)))),2).
weight(P(i(i(x,i(x,y)),i(x,y))),2).
weight(P(j(j(i(x,y),y),j(i(y,x),x))),2).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).
weight(P(j(i(x,i(x,y)),i(x,y))),2).
weight(P(j(j(j(i(x,y),y),i(y,x)),j(j(i(x,y),y),x))),2).
weight(P(i(i(i(x,y),i(z,x)),i(i(x,y),i(z,y)))),2).
weight(P(j(i(i(x,y),x),x)),2).
weight(P(j(j(i(i(x,y),y),y),i(x,y))),2).
weight(P(j(i(i(x,y),i(z,x)),i(i(x,y),i(z,y)))),2).
weight(P(i(i(j(x,y),z),i(y,z))),2).
weight(P(i(x,j(i(x,y),y))),2).
weight(P(i(j(x,y),j(j(z,x),j(z,y)))),2).
weight(P(i(i(x,j(y,z)),i(x,j(j(u,y),j(u,z))))),2).
weight(P(j(j(x,j(i(i(y,z),z),z)),j(x,i(y,z)))),2).
weight(P(j(j(x,i(i(y,z),y)),j(x,y))),2).
weight(P(j(j(x,i(y,i(z,u))),j(x,i(i(y,z),i(y,u))))),2).
weight(P(j(j(x,y),j(x,i(z,y)))),2).
weight(P(i(x,j(j(y,i(x,z)),j(y,z)))),2).
weight(P(i(j(x,j(y,z)),j(j(u,j(x,y)),j(u,j(x,z))))),2).
weight(P(j(i(i(j(x,y),z),y),j(x,y))),2).
weight(P(j(j(x,i(i(y,j(j(z,i(y,u)),j(z,u))),v)),j(x,v))),2).
weight(P(j(j(x,i(i(j(y,z),j(j(u,y),j(u,z))),v)),j(x,v))),2).
weight(P(j(j(x,j(i(i(j(y,z),u),z),y)),j(x,j(i(i(j(y,z),u),z),z)))),2).
weight(P(j(x,j(i(i(j(x,y),z),y),y))),2).
weight(P(j(x,i(j(x,y),y))),2).
weight(P(j(x,i(y,i(j(x,z),z)))),2).
weight(P(j(x,i(i(y,j(x,z)),i(y,z)))),2).
weight(P(j(j(x,y),i(j(y,z),j(x,z)))),2).
weight(P(j(j(x,i(y,z)),i(y,j(x,z)))),2).
weight(P(i(j(i(x,i(j(y,z),z)),u),j(y,u))),2).
weight(P(i(j(j(x,y),z),j(i(x,y),z))),2).
weight(P(j(j(x,j(y,i(z,u))),j(x,i(z,j(y,u))))),2).
weight(P(j(x,i(i(y,z),i(j(x,y),z)))),2).
weight(P(j(i(x,y),i(j(y,z),j(x,z)))),2).
weight(P(i(i(x,y),j(z,i(j(z,x),y)))),2).
weight(P(j(i(x,y),j(z,i(j(z,x),y)))),2).
weight(P(j(i(x,y),i(j(z,x),j(z,y)))),2).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).              % Modus
-P(j(x,y)) | -P(x) | P(y).              % Modus
%  -P(i(i(A,B),j(A,B))) |
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).       % Lemmas
end_of_list.
```

```
list(sos).
%
% Axioms
%
P(i(x,i(y,x))).                     % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).   % (A2)
%  P(i(i(i(x,y),x),x)).              % (A3)
P(i(x,j(y,x))).                     % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).   % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).      % (A6)
%  P(i(j(j(x,y),x),x)).              % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).       % (A8)
P(j(i(x,y),j(x,y))).                 % (A9)
end_of_list.

list(passive).
-P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).                % Lemma
-P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).       % Lemma
-P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).       % Lemma
end_of_list.

list(hot).
-P(i(x,y)) | -P(x) | P(y).              % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

Thus you see why simply finding shorter proofs of members of a disjunction is not a necessarily profitable way to finding a shorter proof of the entire conjunction. Similarly, if the target is a single formula or equation, finding shorter proofs of intermediate steps does not guarantee finding a shorter proof of the target. Distantly related to this discussion is the case in which you have found a rather satisfactory short proof but wish to make more progress, if such is possible, and the study has also led to a different proof of somewhat greater length that appears to merit consideration. To avoid simply reproducing the short proof in hand, you might select a deduced step of the short proof that is not in the longer and add a demodulator that blocks it from being retained. With that move, the program will be unable to complete the same short proof and might find a different proof of the same length or shorter, which can in turn lead to further progress. Alternatively, or perhaps in addition, you could select a deduced step in the longer proof and never block it from being used when iterating with blocking proof steps one at a time.

5.4. Program Latitude and Shorter Proofs

Now, on the subject of finding a shorter proof through giving more latitude, the following input file shows what can occur. In that file, I used a 45-step proof of the join of theses 2 and 3 as hints, a proof that has variable richness equal to 4. However, as you see, I assigned 5, rather than 4, to max_distinct_vars, enabling OTTER to provide what was needed to then complete a 44-step proof of the join, one of richness 5.

An Input File with More Latitude

```
set(hyper_res).
assign(max_weight,23).
%  assign(change_limit_after,1500).
%  assign(new_max_weight,19).
clear(print_kept).
% clear(for_sub).
```

```
set(ancestor_subsume).
set(back_sub).
% clear(set_sub).
assign(max_mem,600000).
%  assign(max_seconds,2).
% set(control_memory).
% assign(report,900).
%  assign(pick_given_ratio,4).
assign(max_proofs,-1).
%set(order_history).
%set(input_sos_first).
set(sos_queue).
%set(print_level).
set(order_history).
%  set(process_input).
assign(max_distinct_vars,5).
assign(heat,0).
assign(bsub_hint_wt,2).
set(keep_hint_subsumers).

%

%
% Clauses
%

list(demodulators).
(P(i(i(i(x,y),x),x)) = junk).  %  A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk).  %  A6
(P(i(j(j(x,y),x),x)) = junk).  %  A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

weight_list(pick_and_purge).
weight(i(i(i(i($(1),$(1)),$(1)),$(1)),$(1)),100).
weight(j(j(j(j($(1),$(1)),$(1)),$(1)),$(1)),100).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).              % Modus
-P(j(x,y)) | -P(x) | P(y).              % Modus
%  -P(i(i(A,B),j(A,B))) |
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).        % Lemmas
end_of_list.

list(sos).
%
% Axioms
%
```

```
P(i(x,i(y,x))).                    % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).   % (A2)
%  P(i(i(i(x,y),x),x)).                % (A3)
P(i(x,j(y,x))).                    % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).   % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).       % (A6)
%  P(i(j(j(x,y),x),x)).                % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).        % (A8)
P(j(i(x,y),j(x,y))).                 % (A9)
%  Following 41 prove thesis 3, may be shortest so far, level 18, vars4, temp.spinks2.join23.out1d37g
P(i(x,i(y,i(z,y)))).
P(j(x,j(y,x))).
P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
P(i(x,i(i(y,i(z,u)),i(i(y,z),i(y,u))))).
P(i(i(x,y),i(x,x))).
P(i(i(x,i(y,i(z,u))),i(x,i(i(y,z),i(y,u))))).
P(i(i(x,i(x,y)),i(x,y))).
P(i(i(x,y),i(i(z,x),i(z,y)))).
P(j(i(x,i(x,y)),i(x,y))).
P(j(i(x,y),i(i(z,x),i(z,y)))).
P(i(i(x,j(y,j(z,u))),i(x,j(j(y,z),j(y,u))))).
P(i(i(x,y),i(x,j(z,y)))).
P(j(j(i(x,y),y),j(i(y,x),x))).
P(i(i(x,j(i(y,z),z)),i(x,j(i(z,y),y)))).
P(j(i(i(x,y),x),x)).
P(i(j(x,y),j(j(z,x),j(z,y)))).
P(i(i(x,y),i(x,i(z,y)))).
P(j(i(x,y),i(x,j(z,y)))).
P(i(x,j(i(x,y),y))).
P(i(i(x,j(y,z)),i(x,j(j(u,y),j(u,z))))).
P(j(j(x,i(i(y,z),y)),j(x,y))).
P(j(j(x,j(i(y,z),z)),j(x,j(i(z,y),y)))).
P(j(j(x,i(y,z)),j(x,i(i(u,y),i(u,z))))).
P(j(i(x,y),i(x,i(z,y)))).
P(i(x,j(j(y,i(x,z)),j(y,z)))).
P(i(j(x,j(y,z)),j(j(u,j(x,y)),j(u,j(x,z))))).
P(j(i(i(j(x,y),z),y),j(x,y))).
P(j(j(x,i(i(y,j(j(z,i(y,u)),j(z,u))),v)),j(x,v))).
P(j(j(x,j(i(i(j(y,z),u),z),y)),j(x,j(i(i(j(y,z),u),z),z)))).
P(j(x,j(i(i(j(x,y),z),y),y))).
P(j(x,j(i(y,i(j(x,y),z)),i(j(x,y),z)))).
P(j(j(x,j(y,i(z,i(j(y,z),u)))),j(x,j(y,i(j(y,z),u))))).
P(j(i(x,i(j(y,x),z)),j(y,i(j(y,x),z)))).
P(j(x,i(j(x,y),y))).
P(j(j(x,i(y,i(j(z,y),u))),j(x,j(z,i(j(z,y),u))))).
P(j(x,i(i(y,j(x,z)),i(y,z)))).
P(j(j(x,i(y,z)),i(y,j(x,z)))).
P(j(x,j(j(y,i(z,u)),i(z,j(y,u))))).
P(j(i(x,y),j(z,i(j(z,x),y)))).
P(j(j(x,j(y,i(z,u))),j(x,i(z,j(y,u))))).
P(j(i(x,y),i(j(z,x),j(z,y)))).
end_of_list.
```

```
list(passive).
% Following negs of a 45/18/4 proof of join of 2 3, from newrose.
-P(j(a1,i(a2,i(a3,a2)))) | $ANS(inter).
-P(j(a1,j(a2,a1))) | $ANS(inter).
-P(i(i(i(a1,i(a2,a3)),i(a1,a2)),i(i(a1,i(a2,a3)),i(a1,a3)))) | $ANS(inter).
-P(i(a1,i(i(a2,i(a3,a4)),i(i(a2,a3),i(a2,a4))))) | $ANS(inter).
-P(i(i(a1,a2),i(a1,a1))) | $ANS(inter).
-P(i(i(a1,i(a2,i(a3,a4))),i(a1,i(i(a2,a3),i(a2,a4))))) | $ANS(inter).
-P(i(i(a1,i(a1,a2)),i(a1,a2))) | $ANS(inter).
-P(i(a1,a1)) | $ANS(inter).
-P(i(i(a1,a2),i(i(a3,a1),i(a3,a2)))) | $ANS(inter).
-P(j(i(a1,i(a1,a2)),i(a1,a2))) | $ANS(inter).
-P(j(i(a1,a2),i(i(a3,a1),i(a3,a2)))) | $ANS(inter).
-P(i(i(a1,j(a2,j(a3,a4))),i(a1,j(j(a2,a3),j(a2,a4))))) | $ANS(inter).
-P(i(i(a1,a2),i(a1,j(a3,a2)))) | $ANS(inter).
-P(j(j(i(a1,a2),a2),j(i(a2,a1),a1))) | $ANS(inter).
-P(i(i(a1,j(i(a2,a3),a3)),i(a1,j(i(a3,a2),a2)))) | $ANS(inter).
-P(j(i(i(a1,a2),a1),a1)) | $ANS(inter).
-P(i(j(a1,a2),j(j(a3,a1),j(a3,a2)))) | $ANS(inter).
-P(j(i(a1,a2),i(a1,j(a3,a2)))) | $ANS(inter).
-P(i(a1,j(i(a1,a2),a2))) | $ANS(inter).
-P(i(i(a1,j(a2,a3)),i(a1,j(j(a4,a2),j(a4,a3))))) | $ANS(inter).
-P(j(j(a1,i(i(a2,a3),a2)),j(a1,a2))) | $ANS(inter).
-P(j(j(a1,j(i(a2,a3),a3)),j(a1,j(i(a3,a2),a2)))) | $ANS(inter).
-P(j(j(a1,i(a2,a3)),j(a1,i(i(a4,a2),i(a4,a3))))) | $ANS(inter).
-P(i(a1,j(j(a2,i(a1,a3)),j(a2,a3)))) | $ANS(inter).
-P(i(j(a1,j(a2,a3)),j(j(a4,j(a1,a2)),j(a4,j(a1,a3))))) | $ANS(inter).
-P(j(i(i(j(a1,a2),a3),a2),j(a1,a2))) | $ANS(inter).
-P(j(a1,i(i(a2,a3),i(a2,i(a4,a3))))) | $ANS(inter).
-P(j(a1,j(j(a2,i(a1,a3)),j(a2,a3)))) | $ANS(inter).
-P(j(j(a1,j(i(i(j(a2,a3),a4),a3),a2)),j(a1,j(i(i(j(a2,a3),a4),a3),a3)))) | $ANS(inter).
-P(j(j(a1,j(a2,j(a3,i(a2,a4)))),j(a1,j(a2,j(a3,a4))))) | $ANS(inter).
-P(j(a1,j(i(i(j(a1,a2),a3),a2),a2))) | $ANS(inter).
-P(j(j(a1,i(a2,a3)),j(a2,j(a1,a3)))) | $ANS(inter).
-P(j(a1,j(i(a2,i(j(a1,a2),a3)),i(j(a1,a2),a3)))) | $ANS(inter).
-P(j(i(a1,a2),j(a3,i(a1,i(a4,a2))))) | $ANS(inter).
-P(j(j(a1,j(a2,i(a3,i(j(a2,a3),a4)))),j(a1,j(a2,i(j(a2,a3),a4))))) | $ANS(inter).
-P(j(i(a1,a2),j(a3,i(j(a3,a1),a2)))) | $ANS(inter).
-P(j(a1,i(j(a1,a2),a2))) | $ANS(inter).
-P(j(a1,i(i(a2,j(a1,a3)),i(a2,a3)))) | $ANS(inter).
-P(j(i(a1,j(a2,a3)),j(a2,i(a1,a3)))) | $ANS(inter).
-P(j(j(a1,i(a2,a3)),i(a2,j(a1,a3)))) | $ANS(inter).
-P(j(j(a1,a2),i(j(a2,a3),j(a1,a3)))) | $ANS(inter).
-P(j(j(a1,j(a2,i(a3,a4))),j(a1,i(a3,j(a2,a4))))) | $ANS(inter).
-P(j(j(a1,j(a2,a3)),j(a1,i(j(a3,a4),j(a2,a4))))) | $ANS(inter).
-P(j(i(a1,a2),i(j(a3,a1),j(a3,a2)))) | $ANS(inter).
-P(j(i(a1,a2),i(j(a2,a3),j(a1,a3)))) | $ANS(inter).
-P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).          % Lemma
-P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).    % Lemma
-P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).    % Lemma
end_of_list.

list(hints).
```

```
% Following 45/18/4 prove join of 2 3, temp.spinks2.join23.in1d25 would give it, 7 not in the 46,
% found 05-19-05.
P(j(x,i(y,i(z,y)))).
P(j(x,j(y,x))).
P(i(i(i(x,i(y,z)),i(x,y)),i(i(x,i(y,z)),i(x,z)))).
P(i(x,i(i(y,i(z,u)),i(i(y,z),i(y,u))))).
P(i(i(x,y),i(x,x))).
P(i(i(x,i(y,i(z,u))),i(x,i(i(y,z),i(y,u))))).
P(i(i(x,i(x,y)),i(x,y))).
P(i(x,x)).
P(i(i(x,y),i(i(z,x),i(z,y)))).
P(j(i(x,i(x,y)),i(x,y))).
P(j(i(x,y),i(i(z,x),i(z,y)))).
P(i(i(x,j(y,j(z,u))),i(x,j(j(y,z),j(y,u))))).
P(i(i(x,y),i(x,j(z,y)))).
P(j(j(i(x,y),y),j(i(y,x),x))).
P(i(i(x,j(i(y,z),z)),i(x,j(i(z,y),y)))).
P(j(i(i(x,y),x),x)).
P(i(j(x,y),j(j(z,x),j(z,y)))).
P(j(i(x,y),i(x,j(z,y)))).
P(i(x,j(i(x,y),y))).
P(i(i(x,j(y,z)),i(x,j(j(u,y),j(u,z))))).
P(j(j(x,i(i(y,z),y)),j(x,y))).
P(j(j(x,j(i(y,z),z)),j(x,j(i(z,y),y)))).
P(j(j(x,i(y,z)),j(x,i(i(u,y),i(u,z))))).
P(i(x,j(j(y,i(x,z)),j(y,z)))).
P(i(j(x,j(y,z)),j(j(u,j(x,y)),j(u,j(x,z))))).
P(j(i(i(j(x,y),z),y),j(x,y))).
P(j(x,i(i(y,z),i(y,i(u,z))))).
P(j(x,j(j(y,i(x,z)),j(y,z)))).
P(j(j(x,j(i(i(j(y,z),u),z),y)),j(x,j(i(i(j(y,z),u),z),z)))).
P(j(j(x,j(y,j(z,i(y,u)))),j(x,j(y,j(z,u))))).
P(j(x,j(i(i(j(x,y),z),y),y))).
P(j(j(x,i(y,z)),j(y,j(x,z)))).
P(j(x,j(i(y,i(j(x,y),z)),i(j(x,y),z)))).
P(j(i(x,y),j(z,i(x,i(u,y))))).
P(j(j(x,j(y,i(z,i(j(y,z),u)))),j(x,j(y,i(j(y,z),u))))).
P(j(i(x,y),j(z,i(j(z,x),y)))).
P(j(x,i(j(x,y),y))).
P(j(x,i(i(y,j(x,z)),i(y,z)))).
P(j(i(x,j(y,z)),j(y,i(x,z)))).
P(j(j(x,i(y,z)),i(y,j(x,z)))).
P(j(j(x,y),i(j(y,z),j(x,z)))).
P(j(j(x,j(y,i(z,u))),j(x,i(z,j(y,u))))).
P(j(j(x,j(y,z)),j(x,i(j(z,u),j(y,u))))).
P(j(i(x,y),i(j(z,x),j(z,y)))).
P(j(i(x,y),i(j(y,z),j(x,z)))).
end_of_list.

list(hot).
-P(i(x,y)) | -P(x) | P(y).              % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

In this study, in the spirit of cramming relying on level saturation, a 41-step proof of thesis 3 was used to find a so-called short proof of thesis 2. A 4-step proof was found. When the two proofs were (so to speak) combined in the next run—although the expectation was the completion of a 45-step proof—OTTER found a 44-step proof of the join of 2 and 3. Such surprises add to the joy of automated reasoning. Conclusion: Allowing OTTER to retain formulas of variable richness 5 rather than 4 shortly led to a small breakthrough, a proof of length 44 to replace one of length 45.

You might say, So what: After all, giving more latitude *should* enable the program to find a shorter proof. Similar observations hold for assignments to max_weight and the like, such as the pick_given_ratio, and in the context of changing the max_weight on the fly. However, you must not conclude that more latitude is always a good thing, either for finding a better proof or for finding a first proof. Keep in mind that, when using breadth first, the extra latitude can drown a program and therefore be impractical.

5.5. Formula Blocking: Some Hints and Caveats

Certain formulas, if blocked from being used, appear to prevent one from completing any proof shorter than that in hand. This remark may seem obvious; but, if you take into account the huge number of proofs that exist for any given theorem, perhaps there is meat in the observation. For example, when attempting to complete a short proof of the conjunction of theses 2 and 3, the following formula when blocked seems to prevent progress.

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

When you encounter this phenomenon, you might unblock the use of such a formula—even if temporary harm occurs in the context of proof length. Also, my experiments suggest that, given the choice, blocking a later formula rather than an earlier formula among the proof steps is more likely to lead to additional progress. An intuitive explanation asserts that the program has more time to replace the use of the later formula. (Of course, remarks about and references to formulas also apply to equations, from what I know.)

Also of interest, experimentation suggests that blocking the first deduced step often prevents further progress toward finding an even shorter proof; such occurs frequently even when the set of axioms contains more than three members, which means that a number of possibilities exist for the first step of a proof. By way of some amusement, if you are studying an area of logic with condensed detachment as the sole rule of inference applied to but one function—as is the case, for example, in various studies of equivalential calculus—and if a single axiom is in use, then blocking the first application of the rule, the first step, prevents the completion of any proof of length 1 or greater. The type of term that is blocked matters much. In that context, you might enjoy studying what occurred in many of my experiments when I blocked the use of double-negation terms.

One can block proof steps one at a time, two at a time, or (sometimes) three at a time. Blocking one step may force the program into a cul de sac, so, in principle, you had best examine the blocking of proof steps in *all* combinations, not a trivial task. Sometimes the way out of a cul de sac is to return to a slightly less pleasing proof and focus on it, an approach that may enable the program to (so to speak) go down a sideroad.

5.6. A Plethora of Choices

In general, the problem of finding a more elegant proof, especially in the context of length, or of finding a first proof in part rests with the number of approaches that can be taken. For example, if you assume that automation is the solution—and it would help—you might consider the choices to make. In particular, you can cram on different subproofs; you can make runs based on different assignments to max_weight, the pick_given_ratio, max_distinct_vars, and such. You can rely on resonators, hints, or both; the choice for resonators or hints has a material effect.

The problem is further complicated because of the interplay of the various parameters and options. It is not their fault; the obstacle rests with the nature of proof search. For example, reducing max_weight might require you to increase the value assigned to max_distinct_vars, and this is but one small taste of the problem. (The max_weight list is used to detect unit conflict and also used in the context of forward subsumption.) In short, a complete examination of all of the possible paths to shortening a given proof, in view of the numerous cited approaches and their interplay, is far more than daunting.

Nevertheless, if you are considering various paths to pursue and if you have two proofs on which to focus, I suggest focusing on the proof in which the difference between level and length is greater.

Insight into the structure of a proof can sometimes be gained by placing the negations of its deduced steps in list(passive), relying on a level-saturation search, and assigning a rather small value to max_weight. You can then see the possible levels of the various deduced steps, keeping in mind that they may be derived in a manner not precisely the same as in the proof in hand. You might find, for example, that the proof breaks into two subproofs, each of length roughly half that of the total proof, with the last step of the entire proof obtained from the last steps of the two subproofs. I had in hand recently—but have not yet relocated it—a proof whose last step was obtained from condensed detachment applied to the preceding two steps. What was interesting is that the proof of the earliest of the two cited steps was a subproof of the proof of the next-to-the-last step of the entire proof.

For those of you who enjoy a challenge, I present here a topic for investigation. Take a proof that you have shortened as much as you believed possible, where the axiom set consists of independent axioms only and possibly the omission of chosen independent axioms (as occurs throughout this essay). Then make a run with the inclusion of the axioms omitted. Next, comment out the adjoined axioms. With persistence, a variety of approaches, and a bit of luck, you might succeed in shortening your original proof.

As an example of adding back in formulas at the axiomatic level, I have in hand a 20-step proof (in contrast to the 24-step proof) of thesis 1, with 3, 6, and 7 as part of the axiom set. (Of course, as shown earlier, axioms 3, 6, and 7 are not needed to obtain the desired proof.)

6. Challenges, Open Questions, and Mysteries

I include in this section challenges and open questions, repeating some offered earlier in the essay. The notation used indicates the type of problem (CH for challenge, OQ for open question), followed by its number (01 is the first), followed by the notebook number (NB1, in this case). Hence, CH02.NB1 is the notation for the second challenge in this technical notebook. I also include mysteries to solve, occurrences and behaviors I cannot explain.

CH01.NB1: The challenge you are offered is to **automate** much of what is detailed in this notebook. In particular, with the knowledge that an exhaustive attack is, in most cases, untenable, you are asked to write a program that takes a proof of a given theorem and eventually returns a shorter proof, ideally a far shorter proof. That program would be useful in the context of what I present in other notebooks.

CH02.NB1: In this challenge, you are asked to extend, modify, and adapt the material presented here for proof shortening to the goal of **proof finding.** I believe that much of the methodology presented in this notebook, as well as other notebooks, can be employed in the discovery of first proofs. Specifically, to complement Veroff's brilliant and powerful sketches technique for proof finding, I am confident that, although the task is difficult, you can extrapolate from what is offered here to produce methodology or methodologies to proof finding. I have in mind the case in which no proof exists to your knowledge, whether the so-called theorem is claimed to be true or not. Conjectures would be one of the targets. For example, in the spirit of cramming, an approach would consist of proving lesser results than the target, lemmas and theorems of less interest, and then have the program rely on their proof steps to complete the proof of the target. (Intuitively, in the cramming strategy, the program is instructed to force, or cram, the deduced steps of a proper subproof into the proof of the theorem as a whole in a manner that such steps are used as parents twice, thrice, or more. An amusing side-effect of a successful use of the cramming strategy is that various proper subproofs of so-called intermediate steps are sometimes traded for longer subproofs of those steps. This phenomenon frequently occurs when proving a conjunction, where the focus is on individual members of the conjunction rather than on intermediate steps. This side effect is acceptable because the goal is to find a shorter proof of the full theorem.) This challenge, if met, might indeed lead to one or more publications of significance.

CH03.NB1: As shorter and still shorter subproofs are found, you are challenged to identify **properties of subproofs** that strongly suggest progress is being made toward the goal of completing a shorter proof of the final result. As noted earlier, shorter subproofs of members of a conjunction or of some intermediate step do not necessarily indicate progress is being made toward the final goal. On the other hand, sometimes shorter subproofs do enable a program or person to complete a shorter total proof. Why do some shorter subproofs prove useful while others (in effect) get in the way? When is it profitable to use a shorter subproof found later after the goal has been reached? Can one predict which shorter subproofs are useful and which not, when the goal is a shorter total proof? As part of this challenge problem, you also are asked to find so-called **metaproperties** that suggest that the actions being taken are wise ones. Blocking the first step of a given proof, for example, can prevent further advance in proof shortening. Similarly, changing the value assigned to max_weight can aid or hinder progress. Are their metaproperties, or properties, of the actions that predict, before the fact, that wise choices are being made?

CH04.NB1: You are challenged to explain why the following input file yields an 18-step proof of the dependence of *A*7 when the ratio strategy is used and when level saturation is not, whereas a 20-step proof is returned when the ratio strategy is avoided but level saturation is used. I have given this problem some thought, but I cannot explain the disparate behaviors, cannot say why the use of level saturation does not lead to an 18-step proof. I conjecture that the answer is readily available—even though I do not have it—and the analysis that solves the puzzle may be of use in uncovering some more of the mystery of proof search.

An Input File for a Puzzling Occurrence

```
set(hyper_res).
assign(max_weight,23).
% assign(change_limit_after,600).
% assign(new_max_weight,15).
clear(print_kept).
set(ancestor_subsume).
set(back_sub).
assign(max_mem,600000).
% assign(max_seconds,2).
% assign(report,900).
assign(pick_given_ratio,4).
assign(max_proofs,-1).
set(input_sos_first).
% set(sos_queue).
set(order_history).
% set(process_input).
assign(max_distinct_vars,3).
assign(bsub_hint_wt,3).
set(keep_hint_subsumers).
assign(heat,0).


weight_list(pick_and_purge).
weight(i(i(i(i($(1),$(1)),$(1)),$(1)),$(1)),100).
weight(j(j(j(j($(1),$(1)),$(1)),$(1)),$(1)),100).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(j(x,y)) | -P(x) | P(y).
-P(j(i(A,B),i(j(B,C),j(A,C)))) | -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_23).
```

```
-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(tba_all).
end_of_list.

list(sos).
P(i(x,i(y,x))).                  % (A1)
P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).   % (A2)
%  P(i(i(i(x,y),x),x)).                % (A3)
P(i(x,j(y,x))).                  % (A4)
P(i(j(x,j(y,z)),j(j(x,y),j(x,z)))).   % (A5)
%  P(i(j(x,j(y,z)),j(y,j(x,z)))).       % (A6)
%  P(i(j(j(x,y),x),x)).                % (A7)
P(i(j(i(x,y),y),j(i(y,x),x))).       % (A8)
P(j(i(x,y),j(x,y))).                 % (A9)
P(i(j(j(x,y),y),j(j(y,x),x))). % A10
end_of_list.

list(passive).
%  Following negs of an 18/8/3 proof of dependence of a7 on near smaller basis.
-P(j(a1,i(a2,j(a3,a2)))) | $ANS(inter).
-P(j(j(i(a1,a2),a2),j(i(a2,a1),a1))) | $ANS(inter).
-P(j(j(a1,j(a2,a3)),j(j(a1,a2),j(a1,a3)))) | $ANS(inter).
-P(j(a1,j(a2,a1))) | $ANS(inter).
-P(j(a1,i(a2,a1))) | $ANS(inter).
-P(j(j(j(i(a1,a2),a2),i(a2,a1)),j(j(i(a1,a2),a2),a1))) | $ANS(inter).
-P(j(j(j(a1,j(a2,a3)),j(a1,a2)),j(j(a1,j(a2,a3)),j(a1,a3)))) | $ANS(inter).
-P(j(j(a1,a2),j(a1,a1))) | $ANS(inter).
-P(j(a1,j(a2,i(a3,a2)))) | $ANS(inter).
-P(j(j(i(j(a1,a2),a2),a2),j(a1,a2))) | $ANS(inter).
-P(j(j(a1,j(a1,a2)),j(a1,a2))) | $ANS(inter).
-P(j(j(a1,a2),j(a1,i(a3,a2)))) | $ANS(inter).
-P(j(j(j(a1,a2),a1),a1)) | $ANS(inter).
-P(j(a1,j(j(j(a2,a3),a2),a2))) | $ANS(inter).
-P(j(j(a1,j(j(a2,a3),a2)),j(a1,a2))) | $ANS(inter).
-P(j(j(j(i(a1,a2),a3),a2),i(a1,a2))) | $ANS(inter).
-P(j(j(i(j(j(a1,a2),a1),a1),a1),a1)) | $ANS(inter).
-P(i(j(j(a1,a2),a1),a1)) | $ANS(inter).
-P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
-P(i(p,i(q,p))) | $ANS(tba_1).
-P(i(i(i(p,q),p),p)) | $ANS(tba_2).
-P(i(i(p,q),i(i(q,r),i(p,r)))) | $ANS(tba_3).
-P(i(j(A,B),i(A,B))) | $ANS(thm).
%  -P(i(j(a1,j(a2,a3)),j(a2,j(a1,a3)))) | $ANS(A6).
%  -P(i(j(j(a1,a2),a1),a1)) | $ANS(a7).
%  -P(i(i(i(a1,a2),a1),a1)) | $ANS(a3).
%  -P(i(i(A,B),j(A,B))) | $ANS(THESIS_1).
%  -P(j(i(A,B),i(j(B,C),j(A,C)))) | $ANS(THESIS_2).
%  -P(j(i(B,C),i(j(A,B),j(A,C)))) | $ANS(THESIS_3).
end_of_list.

list(hints).
%  Following 18/8/3 apparently proved A7 dependent on BCSK+, without 3, 6, and of course 7.
P(j(x,i(y,j(z,y)))).
P(j(j(i(x,y),y),j(i(y,x),x))).
```

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

list(demodulators).
% (P(i(i(x,i(j(y,x),z)),i(x,z))) = junk).
(P(i(i(i(x,y),x),x)) = junk). % A3
(P(i(j(x,j(y,z)),j(y,j(x,z)))) = junk). % A6
% (P(i(j(j(x,y),x),x)) = junk). % A7
(i(x,junk) = junk).
(i(junk,x) = junk).
(j(x,junk) = junk).
(j(junk,x) = junk).
(P(junk) = $T).
end_of_list.

list(hot).
-P(i(x,y)) | -P(x) | P(y).          % Modus
P(i(i(x,y),j(x,y))).
end_of_list.
```

OQ01.NB1: In the *BCSK* logic, with the axiom set consisting of 1, 2, 4, 5, 8, and 9, does there exist a proof of length strictly less than 14 (applications of the two forms of condensed detachment) showing *A*3 to be dependent? For the curious, I cite the fine research of Michael Kinyon for an interesting result in the context of proving *A*3 dependent. Specifically, he has shown that *A*3 can be proved by restricting the set of axioms to 1, 2, 8, and 9. With that restriction, the best proof I know of has length 21.

OQ02.NB1: Does there exist a proof of length strictly less than 24 (applications of the two forms of condensed detachment for the *BCSK* logic) deriving thesis 1 from the system consisting of 1, 2, 4, 5, 8, and 9?

OQ03.NB1: With condensed detachment as the only inference rule, and with the only admissible targets for the completion of a proof one of the fourteen shortest single axioms (other than *XCB*) or the independent 2-basis consisting of symmetry and transitivity, does there exist a proof of length strictly less than 22 that establishes *XCB* to be a single axiom for EC?

MY01.NB1: The following mystery confronts me, a situation that is contrary to intuition. I had a proof of length 252 in which equality was the key relation. I blocked its steps one at a time and made no progress. However, when I blocked (with demodulation) steps two at a time, OTTER returned a 251-step proof *all* of whose steps are among the 252. I would have strongly conjectured that blocking the proof steps one at a time would clearly find a 251-step proof with the given property, if such a proof existed.

Further, the run based on blocking steps two at a time yielded but one such 251-step proof, at least in the first 20,000 combinations. (The program that allows me to conduct such an enormous set of experiments is called otter-loopn; it does not return the proof but, instead, says such exists and, in effect, how to get it.) Can you explain what is happening?

MY02.NB1: I have another mystery to solve, one that Veroff might, given a fair amount of time, well solve if presented to him. Imagine that you have a proof **P** obtained by using a set *A* of hints (in the sense of the strategy Veroff formulated). Next, imagine that various methodologies you apply yield a new proof **Q** shorter than **P**. I would guess that replacing *A* with *B*, where *B* consists of the deduced steps of **Q**, might yield an even shorter proof but, failing that, at least yield **Q**. But I have in hand a study in which such a replacement yields no proof of any type. What goes wrong? How might one proceed wisely?

7. Highlights, Review, and Overview

One of the highlights of the study of the *BCSK* logic reported here was the unexpected discovery of (from what I can learn) new axiom dependencies, dependencies that might prove of interest to various logicians. Specifically, among the nine axioms (given in Section 2) typically used to study *BCSK*, both *A*3 and *A*6 are dependent. My colleague Spinks, who was the sole motivating force for my study of this area of logic and who supplied me with the nine axioms, had not known of these dependencies before OTTER and I entered the picture. Further, they are dependent on axioms 1, 2, 4, 5, 8, and 9. Also unexpected, axiom *A*7 is not only unneeded to establish the cited dependencies, that axiom is also unneeded to prove various theorems I was asked by Spinks to consider. Axiom *A*7 is independent of 1, 2, 4, 5, 8, and 9, proved to me by Z. Ernst with a model he found. A conclusion that might reasonably be drawn says that the variety axiomatized with 1, 2, 4, 5, 8, and 9 might merit serious study. In part because one of the main objects in this notebook is to provide a beginning for those interested in proof shortening, as well as a start for those curious about the use of McCune's reasoning program OTTER, the stories of how the cited dependencies were found will be offered in another notebook, a notebook that focuses on both the *BCSK* logic and on two extensions of that logic. Here you also are treated to a full meal featuring experimentation. You learn, among other things, that proof-shortening methodologies can be profitably used in proof finding, that sometimes experimentation designed with one objective in mind can produce treasure of quite another type.

On a somewhat philosophical note, perhaps some readers will find goals of the sort I discuss in this notebook a bit puzzling. These goals reflect three of my positions, the following, so evident throughout this notebook. First, it is important that we begin to understand the characteristics of the often huge space of conclusions encountered when seeking a proof or the completion of some other assignment. Second, the study of how to shorten proofs is an excellent way to begin accruing data for analysis and gaining insight into the nature of proof spaces. Third, advances in understanding how to shorten proofs may lead to powerful means for finding first proofs.

I believe that some serious readers may tentatively agree with the first two points but may have reservations about the third. For those readers, I propose as a start the problem of finding shorter proofs showing that *XCB* is a single axiom for the area of logic known as equivalential calculus.

I am delighted that Mark Stickel, upon reading my original challenge concerning a proof for *XCB*, was able to shorten the proof from 25 steps to 24 steps. His result is the response of a serious researcher. Because of Stickel's success, I now have a 22-step proof that I offer in another notebook, that focusing on equivalential calculus. (I have shown the proof to both Ulrich and Stickel.) For the record, I would be delighted if someone were able to produce a different proof as short as mine, and, just possibly, produce an even shorter proof. If someone were able to discover a proof of less than 20 deduced steps (applications of condensed detachment), it would almost surely require either remarkable luck or a significant advance in understanding the structure of these large proof spaces. I believe that anyone who makes a serious effort to complete a proof of less than 20 steps will inevitably begin to face issues that we only vaguely understand at this point. I rather strongly suspect that such an individual would have to utilize techniques that relate to those I discuss in this notebook, but it is also possible that someone might succeed using fresh approaches and insights. In either case, the success would surely be a moment worth savoring.

To close this notebook, I turn to some random notes that might add to your understanding and to your interest in research. First, with regard to proofs yielded in the presence of ancestor subsumption, occasionally the proof length cited by OTTER is greater than the actual proof length, the number of deduced clauses not counting duplicates. In particular, with the use of that powerful procedure (due to McCune), a deduced clause may occur more than once in a proof. Therefore, for complete accuracy, a citation of proof length (when ancestor subsumption is in use) requires you to check for duplicates. This topic, as well as copious information on how research can be conducted, is covered in a book I wrote many years ago, *The Automation of Reasoning: An Experimenter's Notebook with OTTER Tutorial*. I myself often forget to check for this peculiarity. The cited book, I must playfully warn, can be dangerous in the sense that I know of one reader whose reading of it changed his research interests dramatically.

As for an approach to seeking shorter proofs, one that I have not explored much, you might experiment with moving items into and out of the initial set of support. Indeed, access to more elements in list(sos) allows the program to explore paths that it might not otherwise explore. On the other hand, access to fewer elements prevents various paths from being followed. In a similar fashion, a different set of resonators and/or hints from that which has been in use can also have a big effect on the number and nature of reasoning paths that are pursued. Closely related, when no hints are employed, is the approach that replaces all resonators with hints; of course, the converse can produce progress. A third approach to seeking new proofs, perhaps shorter than a proof in hand, focuses on the value you choose to assign in such as the following.

```
assign(change_limit_after,400).
assign(new_max_weight,16).
```

The choice of either or both assigned values can have a powerful effect, especially when a level-saturation approach is employed.

Yet one other approach to finding, say, a shorter proof merits mention. That approach focuses on (in effect) enlisting the effort of another researcher. A fine example of this occurrence was cited earlier in this section and discussed in my notebook on equivalential calculus and the success of Mark Stickel. Stickel provided a second fine success in the context of the *BCI* logic when he informed me (by e-mail), in November 2008, of his finding a 10-step proof that deduces, from the so-called Ulrich 3 single axiom, Meredith's first single axiom for this area of logic. To be more accurate, he sent me seven 10-step proofs; the shortest I had found, offered in my notebook on the *BCK* and *BCI* logics, has length 12. Such correspondence is most welcome and indeed stimulating. He had read my notebook on both the *BCK* and *BCI* logics, which caused him to make his study. Stickel has his own program for applying condensed detachment. I cannot at this time detail the precise method Stickel used to discover these new 10-step proofs, which immediately offers you a challenge. Indeed, as a challenge, you are asked to derive from the Ulrich third single axiom (the first of the following two) for the *BCI* logic Meredith's first single axiom, the second of the following two, and obtain a 10-step proof relying solely on condensed detachment.

```
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).    % 1
P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).    %11
```

I plan to follow this notebook with another that focuses on extensions of the *BCSK* logic. That notebook offers a 3-element model showing *A*7 to be independent among the axioms *A*1 through *A*9, a model found by Z. Ernst. In that notebook, in addition to telling the stories about how the dependencies of both *A*3 and *A*6 were found, I offer other discoveries that were new and surprising.

On a radically different topic, I note that proofs found by OTTER can indeed be instructive and edifying in various ways. This view was certainly contrary to diverse minds in the early 1990s, as cited in a *New York Times* article. I cannot help wondering what those who held that view would say upon hearing that, surprising to me and to Spinks, proofs found by OTTER showed us that *A*7 was unneeded for much of our research focusing on proving diverse theorems in the *BCSK* logic. More evidence, if needed, for the value of relying on the assistance of an automated reasoning program is provided by McCune's success in settling (in the affirmative) the Robbins algebra conjecture. Robbins himself was most pleased to learn (from McCune) that, finally, a proof existed showing that the three axioms for a Robbins algebra suffice for an axiomatization of Boolean algebra. And, as discussed in other of my notebooks, Ulrich and I answered the question, open for seven decades, focusing on the formula known as *XCB*; our proof showing that the cited formula is a single axiom completed the search for shortest single axioms for equivalential calculus. Therefore, if you at least dabble in automated reasoning in the context of some area of logic or mathematics, and if you encounter doubters, you have substantial evidence to cite of its usefulness and power.

I suspect that contributing to my delight in seeking shorter proofs than in hand is the following piquant phenomenon, captured by a simple example. Sometimes I am seeking a shorter proof of a conjunction of three targets. Typically, I place in list(passive) the negation of each target and place in list(usable) the negation of the conjunction. The first of the two actions is motivated by the wish to follow the possible progress that is being made and also by the possibility that I might take a proof of one of the members (of the conjunction) and attempt to directly extend that proof, perhaps by using cramming. The piquancy rests with the occasional trading of a rather short proof of a member of the conjunction for a much longer proof while, at the same time, the proof of the entire conjunction that is completed is shorter than expected.

I do indeed welcome the receipt of proofs to shorten, whether from personal work or from the literature. All being equal, my preference is that such be expressed in OTTER notation. (To obtain a copy of OTTER, the following web address is the key: http://www.mcs.anl.gov/AR/otter/.) With OTTER or some other automated reasoning program—or, perhaps, on your own—you might enjoy considering some of the challenges and open questions I offer in Section 6 or elsewhere in this notebook. As for what might come next, I suspect I shall now turn to the completion of a second notebook that also features the *BCSK* logic and extensions of that logic. There I plan to discuss various results that might be termed “surprises for the experts”.