papers/notebooks/algebra
created 07-06-2009
revised 12-10-2009

Exploring Algebra in Search of Elegant Proofs

Larry Wos

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

1. Visiting Old Friends

An old friend is, typically, somebody you knew quite well some time ago—perhaps many years ago—a person with whom enjoyable times were spent. On the other hand, an old friend can be an area of interest from the past, for example, a study once made in search of new results in the form of new proofs. This notebook is about both types of friend. Among the first are W. McCune, R. Veroff, K. Kunen, and R. Overbeek. Although I am not at the moment in close contact with many of these people, still this material presented here brings back wonderful memories. Among the second type of friend are lattices, Moufang loops, groups, and other areas of algebra. Each of these areas, many years ago, occupied my attention, bringing me intense excitement and pleasure; clearly, I am still interested in them. Whereas some of my notebooks found on my website, automatedreasoning.net, focus on finding and proving new theorems—sometimes theorems about new single axioms, for example—here you will experience, as in others of my notebooks, my search for elegant proofs. You will be treated in this notebook to various journeys, journeys that start with a result I obtained years ago—occasionally, many years ago. The end of each journey, if all goes as planned, will focus on a proof more elegant than discovered in the beginning. Among the possible measures of elegance are proof length (number of deduced steps), proof size (total number of symbols among the deduced steps, not counting parentheses and commas), and maximum number of distinct variables present in the deduced steps. The details of each research journey may include commentary focusing on why certain decisions were made and what consequences were expected.

Before you dive into this material, you might wish to know that the studies reported here answered challenges that were offered years ago. So that you can share with me the surprise that I felt after completing much of the research reported here, I note that I did not intend at the start to meet those challenges48, but, instead, had the original goal of illustrating research paths. Further—and this fact produced the cited surprise and even amusement for me—I had totally forgotten about those precise challenges. You will, therefore, be presented with results that at one time were the object of suggested research for the community. In addition, I shall offer here new questions—and yes, I do not know their answers. I shall begin with my recent study of lattice theory.

2. Lattice Theory

You can commence a study of lattice theory, like so many areas of algebra, by merely presenting the axioms and the denial of the theorem to be proved. Algebra offers a sharp contrast to the various areas of logic that were detailed in the preceding chapters. In particular, one need not include a clause to be used

---

This work was supported in part by the U.S. Department of Energy, under Contract DE-AC02-06CH11357.

with an appropriate inference rule to capture condensed detachment. Equality is "understood" by OTTER, built in, and the rule of choice is paramodulation.

One of the standard bases (axiom systems) for this area consists of the following set of six equations, where "v" denotes union or join and "^" denotes intersection or meet.

y ^ x = x ^ y
(x ^ y) ^ z = x ^ (y ^ z)
y v x = x v y
(x v y) v z = x v (y v z)
x ^ (x v y) = x
x v (x ^ y) = x

Many years ago, my colleague McCune, author of the automated reasoning program OTTER, became interested in the problem of finding a single axiom for lattices. His goal was not just some single axiom; after all, McKenzie had already devised a method that produces single axioms. The use of that method typically produces gigantic (in length) single axioms. McCune sought a single axiom of (undefined) "reasonable" length. Through a variety of techniques that keyed on the cited algorithm but incorporated the assistance of OTTER, he began with a single axiom of more than 1,000,000 symbols and eventually found a 79-symbol single axiom. The nature of his approach guaranteed that the result was sufficient; no proof was needed. But, after a gap in time, McCune decided upon a new approach that would filter candidates, yielding equations that were promising. The goal was a far shorter single axiom.

Among the candidates, after one year he found the following promising 29-symbol equation.

(((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7))) = x.

Success would be his when and if a proof of some basis could be found. The (nonstandard) 4-basis he chose as target was the following. (The properties of associativity for union and intersection in the 6-basis present obstacles that require too much CPU time to overcome.)

y v (x^ (y^z)) = y
y^ (x v (y v z)) = y
((x^y) v (y^z)) v y = y
((x v y)^ (y v z))^y = y

He won. Indeed, if one instructs OTTER to rely on the autonomous mode by including set(auto), which automatically sets various options and assigns various values to diverse parameters, a proof of the 4-basis is found quickly. (McCune and R. Padmanabhan answered a number of questions by relying on the autonomous mode, as reported in their monograph, which provides more evidence of the power of OTTER.)

I entered the picture at the request of McCune, who had in hand a proof that derived the 4-basis from his single axiom. His proof was, if memory serves, of length 128 or greater, and he wished to have a far shorter proof, which was my primary interest in those days. Actually, he had two such single axioms, but the focus was on the so-called first, the following.

(((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.

After many experiments, some based on the *cramming strategy*, I presented him with a 50-step proof on March 6, 2002.

My new goal seven years later was to find a proof of length strictly less than 50 applications of paramodulation. Since I had tried hard to find such a proof in 2002, why did I think I could succeed now? I can shed no light on the question other than observing, as Overbeek did, that now was the time for a notebook on algebra. Following is the input file I started with, the one that had enabled OTTER to produce the cited 50-step proof. Note that when a percent sign occurs in a line, anything from that point to the end of the line is a comment.

An Early Input File for the Study of Lattice Theory

```
% There are 4 goals and this gets 5 proofs: the 4 goals and the combined proof.

op(400, xfx, ^).
op(400, xfx, v).

%  set(knuth_bendix).
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
set(ancestor_subsume).
set(back_sub).
%  set(back_unit_deletion).
%  assign(neg_weight,-5).
%  set(print_proof_as_hints).

assign(max_weight,61).
assign(change_limit_after,100).
assign(new_max_weight,31).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
%  assign(max_seconds,40).

weight_list(pick_and_purge).
%  Following are, first 4, less common basis, second 6, more standard basis;
%  to cause immediate focus to get respective conjunctions proved quickly.
weight(y v (x^ (y^z))=y,0).
weight(y^ (x v (y v z))=y,0).
weight(((x^y) v (y^z)) v y=y,0).
weight(((x v y)^ (y v z))^y=y,0).
weight(y ^ x = x ^ y,0).
weight((x ^ y) ^ z = x ^ (y ^ z),0).
weight(x ^ (x v y) = x,0).
weight(y v x = x v y,0).
weight((x v y) v z = x v (y v z),0).
weight(x v (x ^ y) = x,0).
weight(junk,1000).
end_of_list.

list(usable).
x = x.
b v (a^ (b^c))!=b | b^ (a v (b v c))!=b |  ((a^b) v (b^c)) v b!=b |  ((a v b)^ (b v c))^b!=b |
   $ANS(step_all).
b ^ a != a ^ b | (a ^ b) ^ c != a ^ (b ^ c) | a ^ (a v b) != a | b v a != a v b | (a v b) v c != a v (b v c) |
   a v (a ^ b) != a | $ANS(step_all2).
```

```
end_of_list.

list(sos).
(((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.
end_of_list.

list(demodulators).
EQ((x v (((y^ (x v x)) v (z^ (x v x)))^u))^ (v v (x v x))=x v x,junk).
end_of_list.

list(passive).
%  b v (a ^ (b ^ c)) != b | $ANS(L1).
%  b ^ (a v (b v c)) != b | $ANS(L2).
%  ((a ^ b) v (b ^ c)) v b != b | $ANS(L3).
%  ((a v b) ^ (b v c)) ^ b != b | $ANS(L4).
end_of_list.

list(hints).
%  Following 50 prove all, less standard 4-basis, from temp.lattices.singax.out1w2c1
(((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y.
(((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y.
(((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y.
(((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v (((y v y) v (z^y))^u)))^
    (v v y)= ((y v y) v (z^y))^u.
(((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v ((v7 v y)^ (y v v8)))=y.
(((x v y)^y) v (z^y))^ (u v ((v v y)^ (y v w)))=y.
(((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (v v z)=y^z.
(((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y.
(((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y.
(((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
(((x v y)^y) v (z^y))^ (u v y)=y.
(((x v y)^y) v (y v y))^ (z v y)=y.
(x v (((y^ (x v x)) v (z^ (x v x)))^u))^ (v v (x v x))=x v x.
(x v (y^ (x v x)))^ (z v ((u v (x v x))^ ((x v x) v v)))=x v x.
(((x v y)^y) v ((z^y) v (z^y)))^ (u v y)=y.
(x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x.
((x^y) v (x^y))^ (z v y)= (x^y) v (x^y).
((x v y)^y) v ((x v y)^y)=y.
(x^y)^ (z v y)=x^y.
x^ (y v ((z v x)^ (x v u)))=x.
(((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
((x^y) v (z^y))^ (u v y)= (x^y) v (z^y).
((x v y)^y) v (z^y)=y.
x^ (x v y)=x.
((x v y)^y) v ((z^y) v (u^y))=y.
((x v y)^y) v y=y.
x^ (y v (x v z))=x^ (x v z).
x^ (y v (x v z))=x.
(x^x) v x=x.
x^ ((y v x)^ (x v z))=x.
x^ (y v (x^ (x v z)))=x.
x^x=x.
x v x=x.
```

```
(x v y)^y=y.
x^ (y^x)=y^x.
(x v (((y^x) v (z^x))^u))^ (v v x)=x.
((x v y)^ (y v z))^y=y^ ((x v y)^ (y v z)).
(x v (x^y))^ (z v x)=x.
x v (((y^x) v (z^x))^u)=x.
((x v y)^ (y v z))^y=y.
(((x^y) v (z^y)) v (((x^y) v (z^y))^u))^y= (x^y) v (z^y).
(x^y)^x=x^y.
x v (((y^x) v x)^z)=x.
x v ((y^x)^z)=x.
((x^y) v y)^y= (x^y) v y.
x v (y^ (z^x))=x.
x v (y^ (x^z))=x.
(x^y) v y=y.
((x^y) v (z^y)) v y=y.
((x^y) v (y^z)) v y=y.
end_of_list.
```

At this point, you might consider experimenting with the given input file, modifying its options and taking other actions, to see whether you can find a proof more satisfying than the following (obtained with the given input file).

The Cited 50-Step Proof for McCune's First Single Axiom for Lattice Theory

```
----- Otter 3.2c, December 2001 -----
The process was started by wos on jaguar.mcs.anl.gov,
Wed Mar  6 12:30:01 2002
The command was "otter".  The process ID is 2323.
-----> EMPTY CLAUSE at   2.90 sec ----> 3866 [hyper,3728,2,3500,1223,1904] $ANS(step_all).

Length of proof is 50.  Level of proof is 31.

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

2 [] b v (a^ (b^c))!=b|b^ (a v (b v c))!=b| ((a^b) v (b^c)) v b!=b| ((a v b)^ (b v c))^b!=b|$ANS(step_all).
4 [] (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.
57 [para_into,4.1.1.1.2,4.1.1] (((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y.
58 [para_into,57.1.1.2.2,57.1.1] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y.
69 [para_from,58.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y.
85 [para_from,69.1.1,57.1.1.2.2] (((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v
    (((y v y) v (z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u.
88 [para_from,85.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v ((v7 v y)^
    (y v v8)))=y.
95 [para_into,88.1.1.1.2,88.1.1] (((x v y)^y) v (z^y))^ (u v ((v v y)^ (y v w)))=y.
105 [para_into,95.1.1.2.2,95.1.1] (((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (v v z)=y^z.
127 [para_from,105.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y.
162 [para_into,127.1.1.1.2,127.1.1] (((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y.
188 [para_from,162.1.1,127.1.1.2.2] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
189 [para_from,162.1.1,95.1.1.2.2] (((x v y)^y) v (z^y))^ (u v y)=y.
192 [para_from,162.1.1,57.1.1.2.2] (((x v y)^y) v (y v y))^ (z v y)=y.
246 [para_from,192.1.1,95.1.1.1.1] (x v (y^ (x v x)))^ (z v ((u v (x v x))^ ((x v x) v v)))=x v x.
253 [para_into,246.1.1.1.2,192.1.1] (x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x.
```

338 [para_from,253.1.1,188.1.1.1.2] (((x v y)^y) v ((z^y) v (z^y)))^ (u v y)=y.
420 [para_from,338.1.1,253.1.1.2.2] ((x^y) v (x^y))^ (z v y)= (x^y) v (x^y).
446 [para_into,420.1.1,189.1.1,flip.1] ((x v y)^y) v ((x v y)^y)=y.
571 [para_from,446.1.1,105.1.1.1] (x^y)^ (z v y)=x^y.
573 [para_from,446.1.1,189.1.1.1] x^ (y v x)=x.
574 [para_from,446.1.1,95.1.1.1] x^ (y v ((z v x)^ (x v u)))=x.
621 [para_from,573.1.1,188.1.1.1.2] (((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
848 [para_from,621.1.1,574.1.1.2.2] ((x^y) v (z^y))^ (u v y)= (x^y) v (z^y).
898 [para_into,848.1.1,189.1.1,flip.1] ((x v y)^y) v (z^y)=y.
1025 [para_from,898.1.1,574.1.1.2] x^ (x v y)=x.
1064 [para_into,1025.1.1,621.1.1,flip.1] ((x v y)^y) v ((z^y) v (u^y))=y.
1068 [para_into,1025.1.1,162.1.1,flip.1] ((x v y)^y) v y=y.
1075 [para_from,1025.1.1,571.1.1.1] x^ (y v (x v z))=x^ (x v z).
1153 [para_into,1068.1.1.1.1,1068.1.1] (x^x) v x=x.
1163 [para_from,1068.1.1,574.1.1.2.2.1] x^ (y v (x^ (x v z)))=x.
1164 [para_from,1068.1.1,574.1.1.2] x^ ((y v x)^ (x v z))=x.
1165 [para_from,1068.1.1,573.1.1.2] x^x=x.
1223 [para_into,1075.1.2,1025.1.1] x^ (y v (x v z))=x.
1377 [para_from,1165.1.1,1153.1.1.1] x v x=x.
1430 [para_into,1377.1.1,898.1.1,flip.1] (x v y)^y=y.
1465 [para_into,1430.1.1.1,898.1.1] x^ (y^x)=y^x.
1495 [para_from,1430.1.1,188.1.1.1.1] (x v (((y^x) v (z^x))^u))^ (v v x)=x.
1527 [para_into,1465.1.1.2,1164.1.1] ((x v y)^ (y v z))^y=y^ ((x v y)^ (y v z)).
1777 [para_into,1495.1.1.1.2.1,898.1.1] (x v (x^y))^ (z v x)=x.
1798 [para_into,1495.1.1,1025.1.1] x v (((y^x) v (z^x))^u)=x.
1904 [para_into,1527.1.2,1164.1.1] ((x v y)^ (y v z))^y=y.
1974 [para_into,1777.1.1.2,1064.1.1] (((x^y) v (z^y)) v (((x^y) v (z^y))^u))^y= (x^y) v (z^y).
1987 [para_from,1777.1.1,1164.1.1.2] (x^y)^x=x^y.
2164 [para_into,1798.1.1.2.1.2,1430.1.1] x v (((y^x) v x)^z)=x.
2181 [para_into,1798.1.1.2.1,1798.1.1] x v ((y^x)^z)=x.
3150 [para_from,2164.1.1,1163.1.1.2] ((x^y) v y)^y= (x^y) v y.
3222 [para_into,2181.1.1.2,1465.1.1] x v (y^ (z^x))=x.
3320 [para_into,3150.1.1,1430.1.1,flip.1] (x^y) v y=y.
3500 [para_into,3222.1.1.2.2,1987.1.1] x v (y^ (x^z))=x.
3568 [para_into,3320.1.1.1,1974.1.1] ((x^y) v (z^y)) v y=y.
3728 [para_into,3568.1.1.1.2,1987.1.1] ((x^y) v (y^z)) v y=y.
3866 [hyper,3728,2,3500,1223,1904] $ANS(step_all).

If your choice is to (so-to-speak) go it alone for a while, you had best pause immediately, for I am about to tell you how I proceeded.

I was intent, as is so typical of my studies in algebra, to find a proof that was a forward-reasoning proof (containing no negative deduced clauses) and free of demodulation. I noted that the input file placed no restriction on the number of distinct variables to be present in a deduced step, nor did it rely on McCune's ratio strategy. In that strategy, you assign a value to pick_given_ratio, for example, 2, and (with such an assignment) let the program choose for inference-rule initiation two items based on complexity, 1 based on first come first serve, 2, 1, and the like. A glance at the proof shows that exactly one deduced step, the fifth, relies on nine distinct variables. I made but two changes in the given input file: I assigned the value 2 to pick_given ratio, and I assigned the value 8 to max_distinct_vars.

Well, OTTER did find a proof of the nonstandard 4-basis given earlier in this section. Rather than improving on the cited 50-step proof, however, OTTER presented me with a 72-step proof, which was not totally encouraging. In fact, discouragement was my temporary reaction; indeed, the proof required more than 144,000 CPU-seconds to complete. The obvious conclusion was to pursue a different path, at least a bit different.

I began anew, using the cited input file, assigning 11 to max_distinct_vars and 1 to pick_given_ratio. All began well in the sense that OTTER found a 50-step proof, as it turned out identical to the cited 50-step proof from 2002. As for why I considered this acceptable, I merely note that I now had an input file offering more control over the search, a file that lost no ground in the context of proof length. The next move was to add as *resonators* weight templates corresponding to the fifty steps of the proof, followed by resonators corresponding to the elements of the 4-basis, followed by resonators corresponding to elements of the 6-basis. A resonator is a formula or equation whose variables are treated as indistinguishable. Their inclusion directs the program by assigning to any deduced-and-retained new item the same value assigned to the resonator that it matches, treating all variables as indistinguishable. The presence of the so-called basis resonators causes OTTER to focus on each member when and if it is deduced.

With a series of experiments, each designed to find a proof shorter than length 50, OTTER eventually presented me with a 47-step proof. The experiments were based on blocking, with demodulation, the steps of the proof in hand, blocking them one at a time. (I note that I am limiting the story to the highlights of the study in focus.) At this point in the narration, I pause to discuss cramming, since it was used in this study of McCune's monumental contribution to lattice theory.

The basic idea in the cramming strategy is to select a proof, in this case a 47-step proof, and focus on part of it. The chosen part might be the first so many steps, in the case in focus, the first thirty-four. Instead, the chosen part might be a subproof of one of the members to be proved, for example, one of the four members of the 4-basis. The chosen deduced steps are adjoined to list(sos), and the program is instructed to focus on each of them, before focusing on any deduced step, to initiate applications of the chosen inference rule(s). The intention is to have the chosen steps do double duty, triple duty, or more, as parents. The idea is to force the chosen steps into a total proof, a proof that (one hopes) replaces the total proof in hand with a shorter proof. One of the consequences of cramming often is the discovery of new subproofs—say, of the members of a conjunction—by longer subproofs than are present in the proof under study. No price is paid; indeed, the goal is a proof, total proof of, say, the conjunction, that is strictly shorter than the proof in hand, with little or no regard for the lengths of the subproofs of its members.

As noted, I chose the first thirty-four steps of the 47-step proof and had OTTER apply a level-saturation approach. What occurred was the discovery of ten equations that, if considered with the chosen thirty-four, would lead to the presentation of a 44-step proof. Success: a 44-step proof was found. For an update, the 47-step proof contains twelve equations not in the 50-step proof. The 44-step proof also contains twelve equations not in the 50-step proof; the 44-step proof contains two equations not in the 47-step proof. Progress was indeed occurring: This story is a good story.

In place of more detail, I now skip to the last item in this chapter of the unfolding story. With more cramming, a reduction on the assignment to max_weight, and a few small additional changes, OTTER returned the following 42-step proof.

A 42-Step Proof in Lattice Theory

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Wed Jul  8 19:26:14 2009
The command was "otter".  The process ID is 6641.

-----> EMPTY CLAUSE at   4.39 sec ----> 3939 [hyper,3647,2,2395,1144,3103] $ANS(step_all).

Length of proof is 42.  Level of proof is 28.

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

2 [] b v (a^ (b^c))!=b|b^ (a v (b v c))!=b| ((a^b) v (b^c)) v b!=b| ((a v b)^ (b v c))^b!=b|$ANS(step_all).
4 [] (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.
```

13 [para_into,4.1.1.1.2,4.1.1] (((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y.
14 [para_into,13.1.1.2.2,13.1.1] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y.
23 [para_from,14.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y.
38 [para_from,23.1.1,13.1.1.2.2] (((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v (((y v y) v (z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u.
49 [para_from,38.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v ((v7 v y)^ (y v v8)))=y.
56 [para_into,49.1.1.1.2,49.1.1] (((x v y)^y) v (z^y))^ (u v ((v v y)^ (y v w)))=y.
68 [para_into,56.1.1.2.2,56.1.1] (((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (v v z)=y^z.
91 [para_from,68.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y.
114 [para_into,91.1.1.1.2,91.1.1] (((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y.
136 [para_from,114.1.1,91.1.1.2.2] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
137 [para_from,114.1.1,56.1.1.2.2] (((x v y)^y) v (z^y))^ (u v y)=y.
140 [para_from,114.1.1,13.1.1.2.2] (((x v y)^y) v (y v y))^ (z v y)=y.
243 [para_from,140.1.1,56.1.1.1.1] (x v (y^ (x v x)))^ (z v ((u v (x v x))^ ((x v x) v v)))=x v x.
251 [para_into,243.1.1.1.2,140.1.1] (x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x.
327 [para_from,251.1.1,136.1.1.1.2] (((x v y)^y) v ((z^y) v (z^y)))^ (u v y)=y.
418 [para_from,327.1.1,251.1.1.2.2] ((x^y) v (x^y))^ (z v y)= (x^y) v (x^y).
444 [para_into,418.1.1,137.1.1,flip.1] ((x v y)^y) v ((x v y)^y)=y.
575 [para_from,444.1.1,91.1.1.1.2.1] (((x v y)^y) v (y^z))^ (u v ((v v y)^ (y v w)))=y.
579 [para_from,444.1.1,56.1.1.1] x^ (y v ((z v x)^ (x v u)))=x.
716 [para_into,579.1.1.2.2,68.1.2] x^ (y v ((((z v ((u v x)^ (x v v)))^ ((u v x)^ (x v v))) v (w^ ((u v x)^ (x v v))))^ (v6 v (x v v))))=x.
738 [para_from,579.1.1,136.1.1.1.2] (((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
938 [para_from,738.1.1,579.1.1.2.2] ((x^y) v (z^y))^ (u v y)= (x^y) v (z^y).
1111 [para_into,938.1.1,137.1.1,flip.1] ((x v y)^y) v (z^y)=y.
1144 [para_from,1111.1.1,716.1.1.2] x^ (y v (x v z))=x.
1155 [para_from,1111.1.1,575.1.1.2] (((x v y)^y) v (y^z))^ (y v u)=y.
1207 [para_into,1155.1.1,579.1.1] ((x v y)^y) v (y^z)=y.
1282 [para_into,1207.1.1.2,579.1.1] ((x v y)^y) v y=y.
1292 [para_from,1207.1.1,68.1.1.2] (((x v (y^ (z^u)))^ (y^ (z^u))) v (v^ (y^ (z^u))))^z=y^ (z^u).
1314 [para_from,1282.1.1,579.1.1.2] x^ ((y v x)^ (x v z))=x.
1343 [para_from,1282.1.1,1111.1.1.1.1] (x^x) v (y^x)=x.
1454 [para_into,1314.1.1.2,575.1.1] (x^y)^x=x^y.
1460 [para_into,1314.1.1.2,114.1.1] x^x=x.
1619 [para_from,1454.1.1,738.1.1.1.2.2] (((x v y)^y) v ((z^y) v (y^u)))^ (v v y)=y.
1803 [para_from,1460.1.1,1343.1.1.1] x v (y^x)=x.
1823 [para_from,1460.1.1,738.1.1.1.2.2] (((x v y)^y) v ((z^y) v y))^ (u v y)=y.
2045 [para_from,1619.1.1,1314.1.1.2] ((x^y) v (y^z))^y= (x^y) v (y^z).
2393 [para_into,1803.1.1.2,1314.1.1] ((x v y)^ (y v z)) v y= (x v y)^ (y v z).
2395 [para_into,1803.1.1.2,1292.1.1] x v (y^ (x^z))=x.
2415 [para_into,1803.1.1,1207.1.1,flip.1] (x v y)^y=y.
3103 [para_into,2415.1.1.1,2393.1.1] ((x v y)^ (y v z))^y=y.
3123 [para_into,2415.1.1,1823.1.1,flip.1] (x^y) v y=y.
3647 [para_into,3123.1.1.1,2045.1.1] ((x^y) v (y^z)) v y=y.

This 42-step proof exhibits an amusing (to me) property. In particular, all of its deduced steps are among the deduced steps of the 44-step proof. How piquant! The proof contains eleven equations not present in the 50-step proof that was in focus in the beginning. As is true of the original 50-step proof, exactly one deduced step in the 42-step proof relies on nine distinct variables. That fact naturally led to the next chapter in this story.

Elegance of a proof can be measured in various ways. Proof length has been the focus in this tale of lattice theory, so far. Now, the focus is on variable richness. The question to be considered asks about the existence of a proof, for McCune's first (new) single axiom, whose variable richness is 8 or less. That question, as it turned out, was already answered by the 72-step proof I obtained at the beginning of this episode, the following.

An 8-Variable Proof in Lattice Theory

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Mon Jul 6 11:28:54 2009
The command was "otter". The process ID is 19618.

-----> EMPTY CLAUSE at 144480.87 sec ----> 1674647 [hyper,1669259,2,1663195,1609057,
1618427] $ANS(step_all).

Length of proof is 72. Level of proof is 37.

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

2 [] b v (a^ (b^c))!=b|b^ (a v (b v c))!=b| ((a^b) v (b^c)) v b!=b| ((a v b)^ (b v c))^b!=b|$ANS(step_all).
4 [] (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.
56 [para_into,4.1.1.1.2,4.1.1] (((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y.
57 [para_into,56.1.1.2.2,56.1.1] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y.
58 [para_from,56.1.1,4.1.1.2.2] (((x v (y v y))^ (y v y)) v (((z^ ((y v y) v (y v y))) v
(u^ (y v y)))^v))^ (w v y)=y v y.
68 [para_from,57.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y.
92 [para_from,68.1.1,56.1.1.2.2] (((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v
(z^y))^u) v (((y v y) v (z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u.
94 [para_from,92.1.1,58.1.1.1.2.1.1] (((x v (y v y))^ (y v y)) v ((((((y v y) v (y v y)) v
(z^ (y v y)))^u) v (v^ (y v y)))^w))^ (v6 v y)=y v y.
321 [para_into,94.1.1.1.2,68.1.1] (((x v (y v y))^ (y v y)) v (z^ (y v y)))^ (u v y)=y v y.
359 [para_into,321.1.1.1.2,321.1.1] (((x v (y v y))^ (y v y)) v (y v y))^ (z v y)=y v y.
817 [para_into,359.1.1.1.1,359.1.1] ((x v x) v (x v x))^ (y v x)=x v x.
884 [para_from,359.1.1,56.1.1.2.2] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v (y v y))=y v y.
6487 [para_into,884.1.1.1.1,359.1.1] ((x v x) v ((x v x) v (x v x)))^ (y v (x v x))=x v x.
6673 [para_from,884.1.1,359.1.1.1.1] ((x v x) v ((x v x) v (x v x)))^ (y v (x v x))= (x v x) v (x v x).
8581 [para_into,6673.1.1,6487.1.1,flip.1] (x v x) v (x v x)=x v x.
8935 [para_into,8581.1.1.2,8581.1.2] (x v x) v ((x v x) v (x v x))=x v x.
9108 [para_from,8581.1.1,817.1.1.1] (x v x)^ (y v x)=x v x.
9972 [para_from,9108.1.2,8581.1.1.2] (x v x) v ((x v x)^ (y v x))=x v x.
11090 [para_into,8935.1.1.1,817.1.2] (((x v x) v (x v x))^ (y v x)) v ((x v x) v (x v x))=x v x.
14510 [para_from,9972.1.1,56.1.1.2] (((x v y)^y) v (y v y))^ (y v y)=y.
14512 [para_from,9972.1.1,4.1.1.2] (((x v y)^y) v (((z^ (y v y)) v (u^y))^v))^ (y v y)=y.
15744 [para_from,14510.1.1,58.1.1.1.1] (x v (((y^ ((x v x) v (x v x))) v (z^ (x v x)))^u))^ (v v x)=x v x.
151841 [para_from,11090.1.1,56.1.1.1] (x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x.
151928 [para_into,151841.1.1.2.2.1.2,9108.1.2] (x v x)^ (y v ((z v ((x v x)^ (u v x)))^
((x v x) v v)))=x v x.
158136 [para_into,14512.1.1.1.2.1.1,92.1.1] (((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^
(y v y)=y.
268967 [para_into,15744.1.1.1.2.1.1.2,8581.1.1] (x v (((y^ (x v x)) v (z^ (x v x)))^u))^ (v v x)=x v x.
283887 [para_into,268967.1.1.1.2.1.1,92.1.1] (x v (((((x v x) v (y^x))^z) v (u^ (x v x)))^v))^
(w v x)=x v x.
297690 [para_into,158136.1.1.1.2,158136.1.1] (((x v y)^y) v (z^y))^ (y v y)=y.
365454 [para_into,283887.1.1.1.2,68.1.1] (x v (y^x))^ (z v x)=x v x.

390955 [para_from,365454.1.2,8581.1.1.2] (x v x) v ((x v (y^x))^ (z v x))=x v x.
418043 [para_from,390955.1.1,56.1.1.2] (((x v (y^z))^ (y^z)) v ((y^z) v (y^z)))^ (z v z)=y^z.
464040 [para_from,418043.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y.
464041 [para_into,464040.1.1.1.2,464040.1.1] (((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y.
464193 [para_from,464040.1.1,151928.1.1.2.2] ((x^y) v (x^y))^ (z v y)= (x^y) v (x^y).
466167 [para_from,464041.1.1,464040.1.1.2.2] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
466404 [para_into,464193.1.1,297690.1.1,flip.1] ((x v y)^y) v ((x v y)^y)=y.
479473 [para_into,466404.1.1,464193.1.2] (((x v y)^y) v ((x v y)^y))^ (z v y)=y.
479488 [para_from,466404.1.1,466167.1.1.1.2.1] (((x v y)^y) v (y^z))^ (u v y)=y.
479553 [para_from,466404.1.1,297690.1.1.1] x^ (x v x)=x.
489039 [para_from,479553.1.1,466167.1.1.1.2] (((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
1319007 [para_into,479473.1.1.1,466404.1.1] x^ (y v x)=x.
1319107 [para_from,1319007.1.1,464041.1.1.2.2] (((x v y)^y) v y)^ (z v (u v y))=y.
1333078 [para_into,1319107.1.1,1319007.1.1] ((x v y)^y) v y=y.
1333120 [para_into,1333078.1.1.1.1,1333078.1.1] (x^x) v x=x.
1342238 [para_from,1333078.1.1,1319107.1.1.1] x^ (y v (z v x))=x.
1342239 [para_from,1333078.1.1,464041.1.1.1] x^ (y v ((z v x)^ (x v u)))=x.
1358167 [para_from,1333078.1.1,1319007.1.1.2] x^x=x.
1375369 [para_into,1342239.1.1.2.2.1,1333078.1.1] x^ (y v (x^ (x v z)))=x.
1383029 [para_into,1342239.1.1.2.2,489039.1.1] ((x^y) v (z^y))^ (u v y)= (x^y) v (z^y).
1383886 [para_into,1342239.1.1.2,1333078.1.1] x^ ((y v x)^ (x v z))=x.
1384415 [para_from,1358167.1.1,1333120.1.1.1] x v x=x.
1395725 [para_into,1383029.1.1,297690.1.1,flip.1] ((x v y)^y) v (z^y)=y.
1416749 [para_into,1383886.1.1.2,479488.1.1] (x^y)^x=x^y.
1479476 [para_into,1395725.1.1,1384415.1.1] (x v y)^y=y.
1479530 [para_from,1395725.1.1,1342239.1.1.2] x^ (x v y)=x.
1479828 [para_from,1395725.1.1,1342238.1.1.2.2] (x^y)^ (z v y)=x^y.
1489115 [para_into,1479476.1.1.1,1395725.1.1] x^ (y^x)=y^x.
1497977 [para_from,1479476.1.1,479488.1.1.1.1] (x v (x^y))^ (z v x)=x.
1497978 [para_from,1479476.1.1,466167.1.1.1.1] (x v (((y^x) v (z^x))^u))^ (v v x)=x.
1515523 [para_into,1479530.1.1,489039.1.1,flip.1] ((x v y)^y) v ((z^y) v (u^y))=y.
1520935 [para_into,1479828.1.1.1,1479530.1.1] x^ (y v (x v z))=x^ (x v z).
1543569 [para_into,1489115.1.1.2,1383886.1.1] ((x v y)^ (y v z))^y=y^ ((x v y)^ (y v z)).
1600961 [para_into,1497978.1.1,1479530.1.1] x v (((y^x) v (z^x))^u)=x.
1602467 [para_from,1515523.1.1,1497977.1.1.2] (((x^y) v (z^y)) v (((x^y) v (z^y))^u))^y= (x^y) v (z^y).
1609057 [para_into,1520935.1.2,1479530.1.1] x^ (y v (x v z))=x.
1618427 [para_into,1543569.1.2,1383886.1.1] ((x v y)^ (y v z))^y=y.
1619226 [para_into,1600961.1.1.2.1.2,1479476.1.1] x v (((y^x) v x)^z)=x.
1619487 [para_into,1600961.1.1.2.1,1600961.1.1] x v ((y^x)^z)=x.
1637919 [para_from,1619226.1.1,1375369.1.1.2] ((x^y) v y)^y= (x^y) v y.
1639974 [para_into,1619487.1.1.2,1489115.1.1] x v (y^ (z^x))=x.
1641185 [para_into,1637919.1.1,1479476.1.1,flip.1] (x^y) v y=y.
1663195 [para_into,1639974.1.1.2.2,1416749.1.1] x v (y^ (x^z))=x.
1665278 [para_into,1641185.1.1.1,1602467.1.1] ((x^y) v (z^y)) v y=y.
1669259 [para_into,1665278.1.1.1.2,1416749.1.1] ((x^y) v (y^z)) v y=y.

That proof, as you see, contains but one deduced step that relies on eight distinct variables. Therefore, my first foray here in 2009 into lattice theory was, after all, profitable, although (as shown) very much CPU time was required, as was the retention of many new equations. I had assigned the value 8 to max_distinct_vars in the beginning. To obtain the 42-step proof, I assigned the value 11. The 42-step proof has variable richness 9, with but one equation exhibiting this richness. Now you see almost for certain why, at the start, I had difficulty finding the 50-step proof: The assignment of 8 to max_distinct_vars forced the program to search harder and for a long time. The 50-step proof I had found in 2002 contains one equation relying on nine distinct variables. The blocking of that equation, by discarding newly deduced items with richness of 9 or greater, not only blocked the retention of the cited 9-variable equation, but, as a result, presented quite a problem for OTTER—a problem that OTTER did solve.

As you have correctly conjectured, I then wondered whether even more elegance, in the context of variable richness, was obtainable. Specifically, could I find, with OTTER, a proof deducing the 4-basis from McCune's first single axiom, a proof whose variable richness is 7 or less? The goal might indeed be out of reach; after all, the axiom itself relies on eight distinct variables. The key changes I made to the first input file consisted of assigning to max_distinct_vars the value 7 rather than 8, the assignment of the value 1 rather than 2 to pick_given_ratio, and the inclusion (as resonators) of seventy-two items corresponding to the 72-step proof found in the beginning. Of course, the assigned value of 7 was dictated by the goal; the assignment of 1 was motivated by the thought that equations retained early, although perhaps complex, might be useful; the inclusion of the cited resonators was prompted by the fact that the 72-step proof contained but one 8-variable element, and some guidance was assumed to be vital. As shown, the goal of finding a proof deducing the 4-basis with the constraint of variable richness 8 or less proved to be quite an obstacle, even though but one deduced step exhibited variable richness 9 in the 50-step proof found in 2002. Therefore, in that the 72-step proof of variable richness 8 has but one deduced equation with that richness, and in that seventy-two resonators were being used to direct the reasoning, each corresponding to one of the deduced steps of the 72-step proof, seeking a proof of richness 7 might also present a monumental obstacle—perhaps one that would not be overcome.

OTTER did win, as evidenced by the following proof, and, yes, you may indeed find the various numbers impressive.

A Lattice Theory Proof Deducing the 4-Basis with Variable Richness 7

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on octopus.mcs.anl.gov,
Thu Jul  9 15:51:29 2009
The command was "otter".  The process ID is 2320.

-----> EMPTY CLAUSE at 438912.70 sec ----> 3783967 [hyper,3767337,2,3670721,
    3473576,3490804] $ANS(step_all).

Length of proof is 83.  Level of proof is 47.

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

2 [] b v (a^ (b^c))!=b|b^ (a v (b v c))!=b| ((a^b) v (b^c)) v b!=b| ((a v b)^ (b v c))^b!=b|$ANS(step_all).
4 [] (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.
59 [para_into,4.1.1.1.2,4.1.1] (((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y.
60 [para_into,59.1.1.2.2,59.1.1] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y.
61 [para_from,59.1.1,4.1.1.2.2] (((x v (y v y))^ (y v y)) v (((z^ ((y v y) v (y v y))) v
    (u^ (y v y)))^v))^ (w v y)=y v y.
70 [para_from,60.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y.
95 [para_from,70.1.1,59.1.1.2.2] (((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v
    (z^y))^u) v (((y v y) v (z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u.
97 [para_from,95.1.1,61.1.1.1.2.1.1] (((x v (y v y))^ (y v y)) v ((((((y v y) v (y v y)) v
    (z^ (y v y)))^u) v (v^ (y v y)))^w))^ (v6 v y)=y v y.
111 [para_into,97.1.1.1.2,70.1.1] (((x v (y v y))^ (y v y)) v (z^ (y v y)))^ (u v y)=y v y.
128 [para_into,111.1.1.1.2,111.1.1] (((x v (y v y))^ (y v y)) v (y v y))^ (z v y)=y v y.
183 [para_into,128.1.1.1.1,128.1.1] ((x v x) v (x v x))^ (y v x)=x v x.
217 [para_from,128.1.1,59.1.1.2.2] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v (y v y))=y v y.
```

475 [para_into,217.1.1.1.1,128.1.1] ((x v x) v ((x v x) v (x v x)))^ (y v (x v x))=x v x.
510 [para_from,217.1.1,128.1.1.1.1] ((x v x) v ((x v x) v (x v x)))^ (y v (x v x))= (x v x) v (x v x).
820 [para_into,510.1.1,475.1.1,flip.1] (x v x) v (x v x)=x v x.
965 [para_into,820.1.1.2,820.1.2] (x v x) v ((x v x) v (x v x))=x v x.
1016 [para_from,820.1.1,183.1.1.1] (x v x)^ (y v x)=x v x.
1274 [para_into,965.1.1.1,183.1.2] (((x v x) v (x v x))^ (y v x)) v ((x v x) v (x v x))=x v x.
1640 [para_from,1016.1.2,820.1.1.2] (x v x) v ((x v x)^ (y v x))=x v x.
2088 [para_from,1274.1.1,59.1.1.1] (x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x.
2463 [para_from,1640.1.1,59.1.1.2] (((x v y)^y) v (y v y))^ (y v y)=y.
2465 [para_from,1640.1.1,4.1.1.2] (((x v y)^y) v (((z^ (y v y)) v (u^y))^v))^ (y v y)=y.
2722 [para_into,2088.1.1.2.2.1.2,1016.1.2] (x v x)^ (y v ((z v ((x v x)^ (u v x)))^ ((x v x) v v)))=x v x.
3005 [para_from,2463.1.1,111.1.1.1.1] (x v (y^ (x v x)))^ (z v x)=x v x.
3013 [para_from,2463.1.1,61.1.1.1.1] (x v (((y^ ((x v x) v (x v x))) v (z^ (x v x)))^u))^ (v v x)=x v x.
3250 [para_into,3013.1.1.1.2.1.1.2,820.1.1] (x v (((y^ (x v x)) v (z^ (x v x)))^u))^ (v v x)=x v x.
3430 [para_into,3250.1.1.1.2.1.1,95.1.1] (x v (((((x v x) v (y^x))^z) v (u^ (x v x)))^v))^ (w v x)=x v x.
3666 [para_into,3430.1.1.1.2,70.1.1] (x v (y^x))^ (z v x)=x v x.
3835 [para_from,3666.1.2,820.1.1.2] (x v x) v ((x v (y^x))^ (z v x))=x v x.
4102 [para_from,3835.1.1,59.1.1.2] (((x v (y^z))^ (y^z)) v ((y^z) v (y^z)))^ (z v z)=y^z.
4389 [para_from,4102.1.1,2465.1.1.1.2.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (y v y)=y.
4473 [para_into,4389.1.1.1.2,4389.1.1] (((x v y)^y) v y)^ (y v y)=y.
6510 [para_from,4473.1.1,4.1.1.2.2] (((x v y)^y) v (((z^ (y v y)) v (u^y))^v))^ (w v y)=y.
6647 [para_into,6510.1.1.1.2.1.1,4102.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
6651 [para_into,6510.1.1.1.2.1.1,95.1.1] (((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v y)=y.
6737 [para_from,6647.1.1,2722.1.1.2.2] ((x^y) v (x^y))^ (z v y)= (x^y) v (x^y).
6780 [para_into,6651.1.1.1.2,6651.1.1] (((x v y)^y) v (z^y))^ (u v y)=y.
7463 [para_into,6780.1.1,6737.1.1] ((x v y)^y) v ((x v y)^y)=y.
7601 [para_from,7463.1.1,6647.1.1.1.2.1] (((x v y)^y) v (y^z))^ (u v y)=y.
7606 [para_from,7463.1.1,6780.1.1.1] x^ (y v x)=x.
8174 [para_from,7601.1.1,59.1.1.2.2] (((x v (y^z))^ (y^z)) v ((y^z) v (y^z)))^ (u v y)=y^z.
8272 [para_from,7606.1.1,6647.1.1.1.2] (((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
8443 [para_from,8174.1.1,6510.1.1.1.2.1.1] (((x v y)^y) v (((y^z) v (u^y))^v))^ (w v y)=y.
8782 [para_into,8443.1.1.1.2,3430.1.1] (((x v y)^y) v ((y^z) v (y^z)))^ (u v y)=y.
9049 [para_from,8782.1.1,2088.1.1.2.2] ((x^y) v (x^y))^ (z v x)= (x^y) v (x^y).
9099 [para_into,9049.1.1.1,7463.1.1,flip.1] ((x v y)^y) v ((x v y)^y)=y^ (z v (x v y)).
18384 [para_into,3005.1.1.1.2,8174.1.1] (x v (x^y))^ (z v x)=x v x.
536591 [para_into,9099.1.1,7463.1.1,flip.1] x^ (y v (z v x))=x.
541368 [para_into,536591.1.1.2.2,7463.1.1] ((x v y)^y)^ (z v y)= (x v y)^y.
561985 [para_from,541368.1.2,7601.1.1.1.1] ((((x v y)^y)^ (z v y)) v (y^u))^ (v v y)=y.
3102148 [para_from,561985.1.1,4.1.1.1.2] (((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y.
3104848 [para_into,3102148.1.1.2.2,536591.1.1] (((x v y)^y) v y)^ (z v (u v y))=y.
3107253 [para_into,3104848.1.1,7606.1.1] ((x v y)^y) v y=y.
3118388 [para_into,3107253.1.1.1.1,3107253.1.1] (x^x) v x=x.
3119623 [para_from,3107253.1.1,3102148.1.1.1] x^ (y v ((z v x)^ (x v u)))=x.
3120601 [para_from,3107253.1.1,7606.1.1.2] x^x=x.
3146102 [para_into,3119623.1.1.2.2.1,3107253.1.1] x^ (y v (x^ (x v z)))=x.
3147417 [para_into,3119623.1.1.2.2,561985.1.1] (x^y)^ (z v x)=x^y.
3147616 [para_into,3119623.1.1.2.2,8272.1.1] ((x^y) v (z^y))^ (u v y)= (x^y) v (z^y).
3147621 [para_into,3119623.1.1.2.2,6780.1.1] (x^y)^ (z v y)=x^y.
3147633 [para_into,3119623.1.1.2,3107253.1.1] x^ ((y v x)^ (x v z))=x.
3153211 [para_from,3120601.1.1,3118388.1.1.1] x v x=x.
3162209 [para_into,3147417.1.1.2,3107253.1.1] (x^y)^x=x^y.
3175070 [para_into,3147616.1.1,6780.1.1,flip.1] ((x v y)^y) v (z^y)=y.
3185925 [para_into,3153211.1.1,18384.1.2] (x v (x^y))^ (z v x)=x.

3199660 [para_from,3162209.1.1,8272.1.1.1.2.2] (((x v y)^y) v ((z^y) v (y^u)))^ (v v y)=y.
3204487 [para_into,3175070.1.1,3153211.1.1] (x v y)^y=y.
3204511 [para_from,3175070.1.1,3119623.1.1.2] x^ (x v y)=x.
3274813 [para_into,3204487.1.1.1,3175070.1.1] x^ (y^x)=y^x.
3275942 [para_from,3204487.1.1,8443.1.1.1.1] (x v (((x^y) v (z^x))^u))^ (v v x)=x.
3275945 [para_from,3204487.1.1,6647.1.1.1.1] (x v (((y^x) v (z^x))^u))^ (v v x)=x.
3283004 [para_into,3204511.1.1,3199660.1.1,flip.1] ((x v y)^y) v ((z^y) v (y^u))=y.
3284239 [para_from,3204511.1.1,3147621.1.1.1] x^ (y v (x v z))=x^ (x v z).
3329902 [para_into,3274813.1.1.2,3147633.1.1] ((x v y)^ (y v z))^y=y^ ((x v y)^ (y v z)).
3380786 [para_into,3275942.1.1,3204511.1.1] x v (((x^y) v (z^x))^u)=x.
3385916 [para_into,3275945.1.1,3204511.1.1] x v (((y^x) v (z^x))^u)=x.
3429417 [para_from,3283004.1.1,3185925.1.1.2] (((x^y) v (y^z)) v (((x^y) v (y^z))^u))^y= (x^y) v (y^z).
3473576 [para_into,3284239.1.2,3204511.1.1] x^ (y v (x v z))=x.
3490804 [para_into,3329902.1.2,3147633.1.1] ((x v y)^ (y v z))^y=y.
3527366 [para_into,3380786.1.1.2.1,3380786.1.1] x v ((x^y)^z)=x.
3531568 [para_into,3385916.1.1.2.1.2,3204487.1.1] x v (((y^x) v x)^z)=x.
3670721 [para_into,3527366.1.1.2,3274813.1.1] x v (y^ (x^z))=x.
3672697 [para_from,3531568.1.1,3146102.1.1.2] ((x^y) v y)^y= (x^y) v y.
3737025 [para_into,3672697.1.1,3204487.1.1,flip.1] (x^y) v y=y.
3767337 [para_into,3737025.1.1.1,3429417.1.1] ((x^y) v (y^z)) v y=y.

The given proof contains three equations of richness 7. So piquant to me, I cannot at the moment recall having in hand a proof in which four subgoals, the members of the 4-basis, were each reached but only after the retention of more than 3,000,000 new equations. The CPU time that was required, more than 130 CPU-hours, ranks very high in time spent before the final target was in hand. As Overbeek has said in various contexts, ponder this: The finding of a proof to replace that in hand, such that you are merely asked to avoid a single complex equation (of variable richness 8), forced the automated reasoning program to expend a huge effort. But, to a lesser extent, that is what occurred when a proof of variable richness 9 was in hand and the goal was one of richness 8. I find such examples add materially to the intrigue of mathematics and logic; one small modification, on the surface, can present a truly gigantic obstacle. For the curious, I did search for a shorter proof, and I found one of length 74, relying on three deduced equations of richness 7. You might enjoy seeking a proof of this length or perhaps shorter in the given context.

At this point, rather than a focus on a possible proof of richness 6—if such exists—the story turns to the study of McCune's second single axiom, the following.

(((y v x) ^ x) v (((z ^ (x v x)) v (u ^ x)) ^ v)) ^ (((w v x) ^ (v6 v x)) v v7) = x.

A moderate search of my files of seven years ago, coupled with a brisk review of memory, revealed no study on my part of a proof deducing the 4-basis of interest from this second single axiom. Therefore, other than continuing to study the first single axiom in the context of variable richness 6, the natural path led to focusing on the second single axiom that McCune had found many years ago. As expected, I attempted in part to emulate what had worked and in part to build on that success. The following input file served me well, and you might find it useful in re-experiencing what has already been narrated (after making appropriate modifications) or find it useful in the context of new research.

An Input File for Studying McCune's Second Single Axiom for Lattice Theory

```
% There are 4 goals, and this gets 5 proofs: the 4 goals and the combined proof.

op(400, xfx, ^).
op(400, xfx, v).

% set(knuth_bendix).
```

```
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
% set(ancestor_subsume).
set(back_sub).
% set(back_unit_deletion).
% assign(neg_weight,-5).
% set(print_proof_as_hints).

assign(max_weight,71).
assign(change_limit_after,50).
assign(new_max_weight,26).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).
% set(sos_queue).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
assign(max_distinct_vars,11).
assign(pick_given_ratio,1).
% assign(max_seconds,10).

weight_list(pick_and_purge).
% Following 33/26 prove the third member of the four, followed by 10 obtained by cramming on
    the 33, temp.lattices.new.out1m.
weight((((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y,-6).
weight((((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y,-6).
weight((((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y,-6).
weight((((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v
    (((y v y) v (z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u,-6).
weight((((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v ((v7 v y)^ (y v v8)))=y,-6).
weight((((x v y)^y) v (z^y))^ (u v ((v v y)^ (y v w)))=y,-6).
weight((((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (v v z)=y^z,-6).
weight((((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y,-6).
weight((((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y,-6).
weight((((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y,-6).
weight((((x v y)^y) v (z^y))^ (u v y)=y,-6).
weight((((x v y)^y) v (y v y))^ (z v y)=y,-6).
weight((x v (y^ (x v x)))^ (z v ((u v (x v x))^ ((x v x) v v)))=x v x,-6).
weight((x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x,-6).
weight((((x v y)^y) v ((z^y) v (z^y)))^ (u v y)=y,-6).
weight(((x^y) v (x^y))^ (z v y)= (x^y) v (x^y),-6).
weight(((x v y)^y) v ((x v y)^y)=y,-6).
weight((((x v y)^y) v (y^z))^ (u v ((v v y)^ (y v w)))=y,-6).
weight(x^ (y v x)=x,-6).
weight(x^ (y v ((z v x)^ (x v u)))=x,-6).
weight((((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y,-6).
weight(((x^y) v (z^y))^ (u v y)= (x^y) v (z^y),-6).
```

weight(((x v y)^y) v (z^y)=y,-6).
weight((((x v y)^y) v (y^z))^ (y v u)=y,-6).
weight(((x v y)^y) v (y^z)=y,-6).
weight(((x v y)^y) v y=y,-6).
weight(x^ ((y v x)^ (x v z))=x,-6).
weight(x^x=x,-6).
weight((x^x) v (y^x)=x,-6).
weight(x v (y^x)=x,-6).
weight(((x v y)^ (y v z)) v y= (x v y)^ (y v z),-6).
weight((x v y)^y=y,-6).
weight(((x v y)^ (y v z))^y=y,-6).
weight(x^ (y v ((((z v ((u v x)^ (x v v)))^ ((u v x)^ (x v v))) v (w^ ((u v x)^ (x v v))))^
(v6 v (x v v))))=x,-6).
weight(x^ (y v (x v z))=x,-6).
weight((((x v (y^ (z^u)))^ (y^ (z^u))) v (v^ (y^ (z^u))))^z=y^ (z^u),-6).
weight(x v (y^ (x^z))=x,-6).
weight((x^y)^x=x^y,-6).
weight(((x^y) v (z^y))^y= (x^y) v (z^y),-6).
weight((((x v y)^y) v ((z^y) v y))^ (u v y)=y,-6).
weight(((x^y) v ((y^z)^y))^y= (x^y) v (y^z),-6).
weight((x^y) v y=y,-6).
weight(((x^y) v (y^z)) v y=y,-6).
% Following 44/27 prove the theorem, temp.lattices.new.out1h
weight((((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y,-5).
weight((((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y,-5).
weight((((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y,-5).
weight((((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v (((y v y) v
(z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u,-5).
weight((((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v ((v7 v y)^ (y v v8)))=y,-5).
weight((((x v y)^y) v (z^y))^ (u v ((v v y)^ (y v w)))=y,-5).
weight((((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (v v z)=y^z,-5).
weight((((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y,-5).
weight((((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y,-5).
weight((((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y,-5).
weight((((x v y)^y) v (z^y))^ (u v y)=y,-5).
weight((((x v y)^y) v (y v y))^ (z v y)=y,-5).
weight((x v (y^ (x v x)))^ (z v ((u v (x v x))^ ((x v x) v v)))=x v x,-5).
weight((x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x,-5).
weight((((x v y)^y) v ((z^y) v (z^y)))^ (u v y)=y,-5).
weight(((x^y) v (x^y))^ (z v y)= (x^y) v (x^y),-5).
weight(((x v y)^y) v ((x v y)^y)=y,-5).
weight((((x v y)^y) v (y^z))^ (u v ((v v y)^ (y v w)))=y,-5).
weight(x^ (y v x)=x,-5).
weight(x^ (y v ((z v x)^ (x v u)))=x,-5).
weight((((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y,-5).
weight(x^ (y v ((((z v ((u v x)^ (x v v)))^ ((u v x)^ (x v v))) v (w^ ((u v x)^ (x v v))))^
(v6 v (x v v))))=x,-5).
weight(((x^y) v (z^y))^ (u v y)= (x^y) v (z^y),-5).
weight(((x v y)^y) v (z^y)=y,-5).
weight(x^ (y v (x v z))=x,-5).
weight((((x v y)^y) v (y^z))^ (y v u)=y,-5).
weight(((x v y)^y) v (y^z)=y,-5).
weight(((x v y)^y) v y=y,-5).

weight((x^y)^x=x^y,-5).
weight((((x v (y^ (z^u)))^ (y^ (z^u))) v (v^ (y^ (z^u))))^z=y^ (z^u),-5).
weight(x^ ((y v x)^ (x v z))=x,-5).
weight(x^x=x,-5).
weight((x^x) v (y^x)=x,-5).
weight((((x v y)^y) v ((z^y) v (y^u)))^ (v v y)=y,-5).
weight((((x v y)^y) v ((z^y) v y))^ (u v y)=y,-5).
weight(((x^y) v (y^z))^ (u v y)= (x^y) v (y^z),-5).
weight(((x^y) v (y^z))^y= (x^y) v (y^z),-5).
weight(x v (y^x)=x,-5).
weight(((x v y)^ (y v z)) v y= (x v y)^ (y v z),-5).
weight(x v (y^ (x^z))=x,-5).
weight((x v y)^y=y,-5).
weight(((x v y)^ (y v z))^y=y,-5).
weight((x^y) v y=y,-5).
weight(((x^y) v (y^z)) v y=y,-5).
% Following 50/30 has 7 not in an earlier 50 in this run and 9 different from the one in the rose directory.
weight((((x v y)^y) v (y v y))^ (z v ((u v y)^ (y v v)))=y,-2).
weight((((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (z v y)=y v y,-2).
weight((((x v y)^y) v (((y v y) v (z^y))^u))^ (v v ((w v y)^ (y v v6)))=y,-2).
weight((((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v (((y v y) v (z^y))^u)))^ (v v y)= ((y v y) v (z^y))^u,-2).
weight((((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (v6 v ((v7 v y)^ (y v v8)))=y,-2).
weight((((x v y)^y) v (z^y))^ (u v ((v v y)^ (y v w)))=y,-2).
weight((((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (v v z)=y^z,-2).
weight((((x v y)^y) v (((z^y) v (u^y))^v))^ (w v ((v6 v y)^ (y v v7)))=y,-2).
weight((((x v y)^y) v y)^ (z v ((u v y)^ (y v v)))=y,-2).
weight((((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y,-2).
weight((((x v y)^y) v (z^y))^ (u v y)=y,-2).
weight((((x v y)^y) v (y v y))^ (z v y)=y,-2).
weight((x v (y^ (x v x)))^ (z v ((u v (x v x))^ ((x v x) v v)))=x v x,-2).
weight((x v x)^ (y v ((z v (x v x))^ ((x v x) v u)))=x v x,-2).
weight((((x v y)^y) v ((z^y) v (z^y)))^ (u v y)=y,-2).
weight(((x^y) v (x^y))^ (z v y)= (x^y) v (x^y),-2).
weight(((x v y)^y) v ((x v y)^y)=y,-2).
weight((((x v y)^y) v (y^z))^ (u v y)=y,-2).
weight(x^ (y v x)=x,-2).
weight(x^ (y v ((z v x)^ (x v u)))=x,-2).
weight((((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y,-2).
weight(x^ (y v ((((z v ((u v x)^ (x v v)))^ ((u v x)^ (x v v))) v (w^ ((u v x)^ (x v v))))^ (v6 v (x v v))))=x,-2).
weight(((x^y) v (z^y))^ (u v y)= (x^y) v (z^y),-2).
weight(((x v y)^y) v (z^y)=y,-2).
weight(x^ (x v y)=x,-2).
weight((((x v (y^ (z^u)))^ (y^ (z^u))) v (v^ (y^ (z^u))))^u=y^ (z^u),-2).
weight(((x v y)^y) v ((z^y) v (u^y))=y,-2).
weight(((x v y)^y) v y=y,-2).
weight(((x^y) v (z^y))^y= (x^y) v (z^y),-2).
weight((x^x) v x=x,-2).
weight(x^ ((y v x)^ (x v z))=x,-2).
weight(x^x=x,-2).
weight((x^x) v (y^x)=x,-2).

```
weight(x v x=x,-2).
weight((x v y)^y=y,-2).
weight(x^ (y^x)=y^x,-2).
weight(x v ((y^x) v (z^x))=x,-2).
weight(((x v y)^ (y v z))^y=y^ ((x v y)^ (y v z)),-2).
weight(((x v y)^ (y v z))^y=y,-2).
weight(x v (y^x)=x,-2).
weight(x v ((y^x) v x)=x,-2).
weight(((x^y) v y)^y= (x^y) v y,-2).
weight((x^y) v y=y,-2).
weight(((x v y)^y) v (y^z)=y,-2).
weight(x^ (y v (x v z))=x,-2).
weight((x^y)^x=x^y,-2).
weight(x v (y^ (z^x))=x,-2).
weight(x v (y^ (x^z))=x,-2).
weight(((x^y) v (z^y)) v y=y,-2).
weight(((x^y) v (y^z)) v y=y,-2).
% Following are, first 4, less common basis, second 6, more standard basis, to cause immediate focus
    to get respective conjunctions proved quickly.
weight(y v (x^ (y^z))=y,0).
weight(y^ (x v (y v z))=y,0).
weight(((x^y) v (y^z)) v y=y,0).
weight(((x v y)^ (y v z))^y=y,0).
weight(y ^ x = x ^ y,0).
weight((x ^ y) ^ z = x ^ (y ^ z),0).
weight(x ^ (x v y) = x,0).
weight(y v x = x v y,0).
weight((x v y) v z = x v (y v z),0).
weight(x v (x ^ y) = x,0).
weight(junk,1000).
end_of_list.

list(usable).
x = x.
b v (a^ (b^c))!=b | b^ (a v (b v c))!=b |  ((a^b) v (b^c)) v b!=b |  ((a v b)^ (b v c))^b!=b | $ANS(step_all).
b ^ a != a ^ b | (a ^ b) ^ c != a ^ (b ^ c) | a ^ (a v b) != a | b v a != a v b | (a v b) v
    c != a v (b v c) | a v (a ^ b) != a | $ANS(step_all2).
end_of_list.

list(sos).
%  (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (w v ((v6 v x)^ (x v v7)))=x.
(((y v x) ^ x) v (((z ^ (x v x)) v (u ^ x)) ^ v)) ^ (((w v x) ^ (v6 v x)) v v7) = x.
end_of_list.

list(demodulators).
end_of_list.

list(passive).
% Following 19 negs for last 19 of a new for McCune's first 29-letter single axiom for lattices.
    7-step proof.
(a1^a2)^a1!=a1^a2 | $ANS(INTER).
(((a1 v (a2^ (a3^a4)))^ (a2^ (a3^a4))) v (a5^ (a2^ (a3^a4))))^a3!=a2^ (a3^a4) | $ANS(INTER).
a1^ ((a2 v a1)^ (a1 v a3))!=a1 | $ANS(INTER).
```

```
a1^a1!=a1 | $ANS(INTER).
(a1^a1) v (a2^a1)!=a1 | $ANS(INTER).
(((a1 v a2)^a2) v ((a3^a2) v (a2^a4)))^ (a5 v a2)!=a2 | $ANS(INTER).
((a1^a2) v (a2^a3))^ (a4 v a2)!= (a1^a2) v (a2^a3) | $ANS(INTER).
((a1^a2) v (a2^a3))^a2!= (a1^a2) v (a2^a3) | $ANS(INTER).
a1 v (a2^a1)!=a1 | $ANS(INTER).
a1 v ((a2^a1) v (a1^a3))!=a1 | $ANS(INTER).
a1 v (a2^ (a1^a3))!=a1 | $ANS(INTER).
(a1 v a2)^a2!=a2 | $ANS(INTER).
a1 v ((a2^a1) v a1)!=a1 | $ANS(INTER).
((a1^a2) v a2)^a2!= (a1^a2) v a2 | $ANS(INTER).
a1^ (a2^a1)!=a2^a1 | $ANS(INTER).
(a1^a2) v a2!=a2 | $ANS(INTER).
((a1 v a2)^ (a2 v a3))^a2!=a2^ ((a1 v a2)^ (a2 v a3)) | $ANS(INTER).
((a1^a2) v (a2^a3)) v a2!=a2 | $ANS(INTER).
((a1 v a2)^ (a2 v a3))^a2!=a2 | $ANS(INTER).
% Following 4 are to prove to get empty, and prove the theorem.
a^ (b v (a v c))!=a | $ANS(MEMBER).
a v (b^ (a^c))!=a | $ANS(MEMBER).
((a v b)^ (b v c))^b!=b | $ANS(MEMBER).
((a^b) v (b^c)) v b!=b | $ANS(MEMBER).
% b v (a ^ (b ^ c)) != b | $ANS(L1).
% b ^ (a v (b v c)) != b | $ANS(L2).
% ((a ^ b) v (b ^ c)) v b != b | $ANS(L3).
% ((a v b) ^ (b v c)) ^ b != b | $ANS(L4).
end_of_list.

list(hints).
end_of_list.
```

A few comments about the just-given input file might be of use. I commented out ancestor_subsume in part because its use is expensive in CPU time and in part because I wanted to seek a shorter proof; indeed, I was after the first proof I had ever seen for McCune's second axiom. The options I chose for paramodulation are those I typically use; Veroff is my source for these effective choices. The assignment of the value 71 to max_weight was prompted by my suspicion that complex equations might be needed. However, after fifty equations were chosen to initiate the use of paramodulation, I reassigned to max_weight the value 26. This move was prompted by the thought that the program with a much large max_weight might drown in newly retained information. The value 1 assigned to pick_given_ratio was motivated by the thought that equations deduced early that were, at the same time, complex might be needed for the sought-after proof. The assignment of 11 to max_distinct_vars was motivated by the fear that such richness might be necessary. As for guiding OTTER, a glance at the pick_and_purge list shows how the research reported earlier in this section was used, in the form of resonators. And here is a comment worth repeating: Inclusion of resonators corresponding to the members of the conjunction is well advised; indeed, such presence enables OTTER to focus on each when and if deduced, thereby speeding the completion of the proof.

And, as you see from the following, all went well.

The So-Called First Proof for the McCune Second Single Axiom for Lattice Theory

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Fri Aug 21 08:47:25 2009
The command was "otter". The process ID is 414.
```

-----> EMPTY CLAUSE at 769.39 sec ----> 72505 [hyper,69346,2,42339,39063,53487] $ANS(step_all).

Length of proof is 100. Level of proof is 35.

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

2 [] b v (a^ (b^c))!=b|b^ (a v (b v c))!=b| ((a^b) v (b^c)) v b!=b| ((a v b)^ (b v c))^b!=b|$ANS(step_all).
4 [] (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (((w v x)^ (v6 v x)) v v7)=x.
28 [para_into,4.1.1.1.2.1.1,4.1.1] (((x v ((y v z)^ (u v z)))^ ((y v z)^ (u v z))) v ((z v (v^ ((y v z)^ (u v z))))^w))^ (((v6 v ((y v z)^ (u v z)))^ (v7 v ((y v z)^ (u v z)))) v v8)= (y v z)^ (u v z).
29 [para_into,4.1.1.1.2,4.1.1] (((x v y)^y) v (y v y))^ (((z v y)^ (u v y)) v v)=y.
30 [para_into,29.1.1.2.1,29.1.1] (((x v (y v y))^ (y v y)) v ((y v y) v (y v y)))^ (y v z)=y v y.
31 [para_into,29.1.1.2.1,4.1.1] (((x v (((y^ (z v z)) v (u^z))^v))^ (((y^ (z v z)) v (u^z))^v)) v ((((y^ (z v z)) v (u^z))^v) v (((y^ (z v z)) v (u^z))^v)))^ (z v w)= ((y^ (z v z)) v (u^z))^v.
33 [para_from,28.1.2,29.1.1.2.1] (((x v y)^y) v (y v y))^ (((((z v ((u v y)^ (v v y)))^ ((u v y)^ (v v y))) v ((y v (w^ ((u v y)^ (v v y))))^v6))^ (((v7 v ((u v y)^ (v v y)))^ (v8 v ((u v y)^ (v v y)))) v v9)) v v10)=y.
34 [para_into,30.1.1.1.1.1.2,30.1.2] (((x v ((((y v (z v z))^ (z v z)) v ((z v z) v (z v z)))^ (z v u)))^ (z v z)) v ((z v z) v (z v z)))^ (z v v)=z v z.
43 [para_from,30.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((y v y) v (z^y))^u))^ (((v v y)^ (w v y)) v v6)=y.
53 [para_from,31.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((((z^ (y v y)) v (u^y))^v) v (w^y))^v6))^ (((v7 v y)^ (v8 v y)) v v9)=y.
65 [para_from,43.1.1,29.1.1.2.1] (((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v ((((y v y) v (z^y))^u) v (((y v y) v (z^y))^u)))^ (y v v)= ((y v y) v (z^y))^u.
66 [para_from,43.1.1,4.1.1.2.1] (((x v (((y v y) v (z^y))^u))^ (((y v y) v (z^y))^u)) v (((v^ ((((y v y) v (z^y))^u) v (((y v y) v (z^y))^u))) v (w^ (((y v y) v (z^y))^u)))^v6))^ (y v v7)= ((y v y) v (z^y))^u.
103 [para_from,65.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((((y v y) v (z^y))^u) v (v^y))^w))^ (((v6 v y)^ (v7 v y)) v v8)=y.
168 [para_into,34.1.1.1.1.1.2.1.2.2,30.1.2] (((x v ((((y v (z v z))^ (z v z)) v ((z v z) v ((((u v (z v z))^ (z v z)) v ((z v z) v (z v z)))^ (z v v))))^ (z v w)))^ (z v z)) v ((z v z) v (z v z)))^ (z v v6)=z v z.
477 [para_into,103.1.1.1.2,103.1.1] (((x v y)^y) v (z^y))^ (((u v y)^ (v v y)) v w)=y.
596 [para_into,477.1.1.2.1,477.1.1] (((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (z v v)=y^z.
600 [para_into,477.1.1.2.1,33.1.1] (((x v (y v y))^ (y v y)) v (z^ (y v y)))^ (y v u)=y v y.
608 [para_from,477.1.1,29.1.1.2.1] (((x v (y^z))^ (y^z)) v ((y^z) v (y^z)))^ (z v u)=y^z.
724 [para_from,596.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (((w v y)^ (v6 v y)) v v7)=y.
804 [para_into,600.1.1.1.2,600.1.1] (((x v (y v y))^ (y v y)) v (y v y))^ (y v z)=y v y.
806 [para_into,600.1.1.1.2,596.1.1] (((x v (y v y))^ (y v y)) v (z^y))^ (y v u)=y v y.
1368 [para_into,804.1.1.1.1,804.1.1] ((x v x) v (x v x))^ (x v y)=x v x.
1413 [para_from,804.1.1,477.1.1.2.1] (((x v (y v y))^ (y v y)) v (z^ (y v y)))^ ((y v y) v u)=y v y.
1415 [para_from,804.1.1,600.1.1.1.1] ((x v x) v (y^ (x v x)))^ (x v z)=x v x.
2108 [para_into,1415.1.1.1.2,596.1.1] ((x v x) v (y^x))^ (x v z)=x v x.
2112 [para_into,1415.1.1.1.2,31.1.1] ((x v x) v (((y^ (x v x)) v (z^x))^u))^ (x v v)=x v x.
2555 [para_from,2108.1.2,1368.1.1.1.1] ((((x v x) v (y^x))^ (x v z)) v (x v x))^ (x v u)=x v x.
5146 [para_into,2112.1.1.1.2.1.1,596.1.1] ((x v x) v (((y^x) v (z^x))^u))^ (x v v)=x v x.
5551 [para_into,5146.1.1.1.2,53.1.1] ((x v x) v x)^ (x v y)=x v x.
5961 [para_from,5551.1.1,477.1.1.2.1] (((x v y)^y) v (z^y))^ ((y v y) v u)=y.
5977 [para_from,5551.1.1,4.1.1.2.1] (((x v y)^y) v (((z^ (y v y)) v (u^y))^v))^ ((y v y) v w)=y.
6331 [para_from,5961.1.1,1415.1.1.1.2] (((x v x) v (x v x)) v x)^ ((x v x) v y)= (x v x) v (x v x).
6353 [para_from,6331.1.2,1368.1.1.1] ((((x v x) v (x v x)) v x)^ ((x v x) v y))^ (x v z)=x v x.
6354 [para_into,1413.1.1.1.1,2555.1.1] ((x v x) v (y^ (x v x)))^ ((x v x) v z)=x v x.
6360 [para_into,6354.1.1.1.2,6353.1.1] ((x v x) v (x v x))^ ((x v x) v y)=x v x.
6364 [para_into,6354.1.1.1.2,66.1.1] ((x v x) v (((x v x) v (y^x))^z))^ ((x v x) v u)=x v x.

6403 [para_into,5977.1.1.1.2.1.1,608.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ ((y v y) v w)=y.
6404 [para_into,6403.1.1.1.2,6403.1.1] (((x v y)^y) v y)^ ((y v y) v z)=y.
6418 [para_from,6404.1.1,477.1.1.2.1] (((x v y)^y) v (z^y))^ (y v u)=y.
6420 [para_from,6404.1.1,29.1.1.2.1] (((x v y)^y) v (y v y))^ (y v z)=y.
6428 [para_into,6418.1.1.1.2,6418.1.1] (((x v (y v z))^ (y v z)) v y)^ ((y v z) v u)=y v z.
6463 [para_from,6420.1.1,806.1.1.1.1] (x v (y^x))^ (x v z)=x v x.
6464 [para_from,6420.1.1,804.1.1.1.1] (x v (x v x))^ (x v y)=x v x.
6471 [para_from,6420.1.1,6418.1.1.1.1] (x v (y^ (x v x)))^ ((x v x) v z)=x v x.
6489 [para_into,6463.1.1,6364.1.1,flip.1] (x v x) v (x v x)=x v x.
6573 [para_into,6489.1.1.2,6489.1.2] (x v x) v ((x v x) v (x v x))=x v x.
6638 [para_from,6489.1.1,6360.1.1.1] (x v x)^ ((x v x) v y)=x v x.
6659 [para_from,6489.1.1,1368.1.1.1] (x v x)^ (x v y)=x v x.
6661 [para_from,6489.1.1,6404.1.1.2] (((x v y)^y) v y)^ (y v y)=y.
6893 [para_from,6659.1.2,6418.1.1.2] (((x v y)^y) v (z^y))^ ((y v y)^ (y v u))=y.
7394 [para_into,6573.1.1.1,6464.1.2] ((x v (x v x))^ (x v y)) v ((x v x) v (x v x))=x v x.
8152 [para_into,6471.1.1.1.2,168.1.1] (x v (x v x))^ ((x v x) v y)=x v x.
8161 [para_into,6471.1.1.1.2,608.1.1] (x v (y^x))^ ((x v x) v z)=x v x.
8369 [para_from,8152.1.2,6661.1.1.1.1.1] ((((x v (x v x))^ ((x v x) v y))^x) v x)^ (x v x)=x.
8459 [para_from,8161.1.2,6638.1.1.2.1] (x v x)^ (((x v (y^x))^ ((x v x) v z)) v u)=x v x.
12948 [para_from,6428.1.1,477.1.1.2.1] (((x v y)^y) v (z^y))^ ((y v u) v v)=y.
14087 [para_from,7394.1.1,29.1.1.1] (x v x)^ (((y v (x v x))^ (z v (x v x))) v u)=x v x.
14158 [para_into,724.1.1.1.2,724.1.1] (((x v y)^y) v y)^ (((z v y)^ (u v y)) v v)=y.
14160 [para_into,724.1.1.2.1,8369.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (y v w)=y.
14167 [para_into,14160.1.1.1.2,8459.1.1] (((x v y)^y) v ((z^y) v (z^y)))^ (y v u)=y.
14353 [para_into,14158.1.1.2.1,6428.1.1] (((x v y)^y) v y)^ ((y v z) v u)=y.
14781 [para_into,14087.1.1.2.1,14167.1.1] ((x^y) v (x^y))^ (y v z)= (x^y) v (x^y).
14805 [para_into,14781.1.1,6418.1.1,flip.1] ((x v y)^y) v ((x v y)^y)=y.
15169 [para_from,14805.1.1,14160.1.1.1.2.1] (((x v y)^y) v (y^z))^ (y v u)=y.
15179 [para_from,14805.1.1,12948.1.1.1] x^ ((x v y) v z)=x.
15184 [para_from,14805.1.1,6893.1.1.1] x^ ((x v x)^ (x v y))=x.
15187 [para_from,14805.1.1,6418.1.1.1] x^ (x v y)=x.
15189 [para_from,14805.1.1,477.1.1.1] x^ (((y v x)^ (z v x)) v u)=x.
15306 [para_from,15169.1.1,477.1.1.2.1] (((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (y v v)=y^z.
16226 [para_from,15184.1.1,14160.1.1.1.2] (((x v y)^y) v ((z^y) v (u^y)))^ (y v v)=y.
16573 [para_into,15306.1.1.1,14805.1.1] (x^y)^ (x v z)=x^y.
23059 [para_into,16573.1.2,15187.1.1] (x^ (x v y))^ (x v z)=x.
30611 [para_into,15189.1.1.2.1,15189.1.1] x^ ((y v x) v z)=x.
30616 [para_into,15189.1.1.2.1,16226.1.1] ((x^y) v (z^y))^ (y v u)= (x^y) v (z^y).
30693 [para_into,30616.1.1,6418.1.1,flip.1] ((x v y)^y) v (z^y)=y.
31316 [para_into,30611.1.1,14353.1.1,flip.1] ((x v y)^y) v y=y.
31332 [para_into,31316.1.1.1.1,31316.1.1] (x^x) v x=x.
31780 [para_from,31316.1.1,30693.1.1.1.1] (x^x) v (y^x)=x.
31899 [para_from,31332.1.1,15189.1.1.2] x^ (y v x)=x.
31902 [para_from,31332.1.1,724.1.1.2] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
32209 [para_into,31899.1.1.2,31332.1.1] x^x=x.
32217 [para_into,31899.1.1,15169.1.1,flip.1] ((x v y)^y) v (y^z)=y.
32432 [para_into,31902.1.1.1.2,31899.1.1] (((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
33212 [para_from,32209.1.1,31780.1.1.1] x v (y^x)=x.
33668 [para_from,32217.1.1,31899.1.1.2] (x^y)^x=x^y.
34014 [para_into,32432.1.1.1.2.2,32209.1.1] (((x v y)^y) v ((z^y) v y))^ (u v y)=y.
34091 [para_into,33212.1.1.2,23059.1.1] (x v y) v x=x v y.
34170 [para_into,33212.1.1,32217.1.1,flip.1] (x v y)^y=y.
34175 [para_from,33212.1.1,15189.1.1.2] x^ ((y v x)^ (z v x))=x.

34702 [para_from,33212.1.1,30616.1.1.2] ((x^y) v (z^y))^y= (x^y) v (z^y).
35707 [para_into,34014.1.1.1.2.1,33668.1.1] (((x v y)^y) v ((y^z) v y))^ (u v y)=y.
36270 [para_into,34170.1.1,34014.1.1,flip.1] (x^y) v y=y.
37068 [para_from,34175.1.1,33212.1.1.2] ((x v y)^ (z v y)) v y= (x v y)^ (z v y).
37112 [para_into,34702.1.2.2,33668.1.1] ((x^y) v ((y^z)^y))^y= (x^y) v (y^z).
37719 [para_into,35707.1.1,34170.1.1] (x^y) v x=x.
38321 [para_into,36270.1.1.1,35707.1.1] x v (y v x)=y v x.
38701 [para_from,37068.1.1,34170.1.1.1] ((x v y)^ (z v y))^y=y.
39063 [para_from,37112.1.1,36270.1.1.1] ((x^y) v (y^z)) v y=y.
40043 [para_from,37719.1.1,596.1.1.2] (((x v (y^ (z^u)))^ (y^ (z^u))) v (v^ (y^ (z^u))))^z=y^ (z^u).
42339 [para_from,40043.1.1,33212.1.1.2] x v (y^ (x^z))=x.
53487 [para_from,34091.1.1,38701.1.1.1.2] ((x v y)^ (y v z))^y=y.
69346 [para_from,38321.1.1,15179.1.1.2] x^ (y v (x v z))=x.
72505 [hyper,69346,2,42339,39063,53487] $ANS(step_all).

Of course, I did pursue a shorter proof than that of length 100. I eventually found the following, indeed a proof of note because of its length.

A 65-Step Proof for McCune's Second Single Axiom

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Mon Aug 10 05:35:31 2009
The command was "otter".  The process ID is 32671.

-----> EMPTY CLAUSE at   0.39 sec ----> 181 [hyper,178,2,170,159,147] $ANS(step_all).

Length of proof is 65.  Level of proof is 28.

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

2 [] b v (a^ (b^c))!=b|b^ (a v (b v c))!=b| ((a^b) v (b^c)) v b!=b| ((a v b)^ (b v c))^b!=b|$ANS(step_all).
4 [] (((y v x)^x) v (((z^ (x v x)) v (u^x))^v))^ (((w v x)^ (v6 v x)) v v7)=x.
70 [para_into,4.1.1.1.2,4.1.1] (((x v y)^y) v (y v y))^ (((z v y)^ (u v y)) v v)=y.
71 [para_into,70.1.1.2.1,4.1.1] (((x v (((y^ (z v z)) v (u^z))^v))^ (((y^ (z v z)) v (u^z))^v)) v
   ((((y^ (z v z)) v (u^z))^v) v (((y^ (z v z)) v (u^z))^v)))^ (z v w)= ((y^ (z v z)) v (u^z))^v.
72 [para_from,71.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((((z^ (y v y)) v (u^y))^v) v (w^y))^v6))^
   (((v7 v y)^ (v8 v y)) v v9)=y.
73 [para_into,72.1.1.1.2,72.1.1] (((x v y)^y) v (z^y))^ (((u v y)^ (v v y)) v w)=y.
74 [para_into,73.1.1.2.1,73.1.1] (((x v (y^z))^ (y^z)) v (u^ (y^z)))^ (z v v)=y^z.
75 [para_into,73.1.1.2.1,70.1.1] (((x v (y v y))^ (y v y)) v (z^ (y v y)))^ (y v u)=y v y.
76 [para_from,74.1.1,4.1.1.1.2.1.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (((w v y)^ (v6 v y)) v v7)=y.
77 [para_into,75.1.1.1.2,75.1.1] (((x v (y v y))^ (y v y)) v (y v y))^ (y v z)=y v y.
78 [para_into,75.1.1.1.2,74.1.1] (((x v (y v y))^ (y v y)) v (z^y))^ (y v u)=y v y.
79 [para_into,76.1.1.1.2,76.1.1] (((x v y)^y) v y)^ (((z v y)^ (u v y)) v v)=y.
80 [para_from,79.1.1,76.1.1.2.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ (y v w)=y.
81 [para_from,79.1.1,73.1.1.2.1] (((x v y)^y) v (z^y))^ (y v u)=y.
82 [para_from,79.1.1,70.1.1.2.1] (((x v y)^y) v (y v y))^ (y v z)=y.
83 [para_into,81.1.1.1.1.1,77.1.1] ((x v x) v (y^ (x v x)))^ ((x v x) v z)=x v x.
84 [para_into,81.1.1.1.1.2,81.1.1] (((x v (y v z))^ (y v z)) v y)^ ((y v z) v u)=y v z.
85 [para_from,82.1.1,78.1.1.1.1] (x v (y^x))^ (x v z)=x v x.
86 [para_from,82.1.1,77.1.1.1.1] (x v (x v x))^ (x v y)=x v x.
87 [para_into,84.1.1.1.1.1,82.1.1] (x v x)^ ((x v x) v y)=x v x.
```

88 [para_from,84.1.1,76.1.1.2.1] (((x v y)^y) v (((z^y) v (u^y))^v))^ ((y v w) v v6)=y.
89 [para_from,84.1.1,73.1.1.2.1] (((x v y)^y) v (z^y))^ ((y v u) v v)=y.
90 [para_into,85.1.1,83.1.1,flip.1] (x v x) v (x v x)=x v x.
91 [para_from,87.1.1,80.1.1.1.2] (((x v y)^y) v ((z^y) v (z^y)))^ (y v u)=y.
92 [para_into,90.1.1.2,90.1.2] (x v x) v ((x v x) v (x v x))=x v x.
93 [para_into,92.1.1.1,86.1.2] ((x v (x v x))^ (x v y)) v ((x v x) v (x v x))=x v x.
94 [para_from,93.1.1,70.1.1.1] (x v x)^ (((y v (x v x))^ (z v (x v x))) v u)=x v x.
95 [para_into,94.1.1.2.1,91.1.1] ((x^y) v (x^y))^ (y v z)= (x^y) v (x^y).
96 [para_into,95.1.1,81.1.1,flip.1] ((x v y)^y) v ((x v y)^y)=y.
97 [para_from,96.1.1,88.1.1.1.2.1] (((x v y)^y) v (y^z))^ ((y v u) v v)=y.
98 [para_from,96.1.1,89.1.1.1] x^ ((x v y) v z)=x.
99 [para_from,96.1.1,81.1.1.1] x^ (x v y)=x.
100 [para_from,96.1.1,73.1.1.1] x^ (((y v x)^ (z v x)) v u)=x.
104 [para_from,99.1.1,79.1.1.2.1] (((x v y)^y) v y)^ ((z v y) v u)=y.
105 [para_into,100.1.1.2.1,100.1.1] x^ ((y v x) v z)=x.
107 [para_into,104.1.1,99.1.1] ((x v y)^y) v y=y.
108 [para_into,105.1.1,97.1.1,flip.1] ((x v y)^y) v (y^z)=y.
109 [para_into,105.1.1,89.1.1,flip.1] ((x v y)^y) v (z^y)=y.
110 [para_into,107.1.1.1.1,107.1.1] (x^x) v x=x.
111 [para_into,109.1.1.1.1,107.1.1] (x^x) v (y^x)=x.
112 [para_from,110.1.1,100.1.1.2] x^ (y v x)=x.
113 [para_from,110.1.1,76.1.1.2] (((x v y)^y) v (((z^y) v (u^y))^v))^ (w v y)=y.
114 [para_into,111.1.1.2,100.1.1] ((((x v y)^ (z v y)) v u)^ (((x v y)^ (z v y)) v u)) v y= ((x v y)^ (z v y)) v u.
115 [para_into,112.1.1.2,110.1.1] x^x=x.
116 [para_into,112.1.1.2,108.1.1] (x^y)^x=x^y.
119 [para_into,113.1.1.1.2,99.1.1] (((x v y)^y) v ((z^y) v (u^y)))^ (v v y)=y.
120 [para_from,115.1.1,111.1.1.1] x v (y^x)=x.
121 [para_from,115.1.1,110.1.1.1] x v x=x.
122 [para_into,119.1.1.1.2.2,116.1.1] (((x v y)^y) v ((z^y) v (y^u)))^ (v v y)=y.
123 [para_into,119.1.1.1.2.2,115.1.1] (((x v y)^y) v ((z^y) v y))^ (u v y)=y.
125 [para_into,120.1.1.2,99.1.1] (x v y) v x=x v y.
128 [para_into,121.1.1,109.1.1,flip.1] (x v y)^y=y.
129 [para_into,121.1.1,85.1.2] (x v (y^x))^ (x v z)=x.
130 [para_into,122.1.1,105.1.1] ((x v y)^y) v ((z^y) v (y^u))=y.
133 [para_into,128.1.1.1,114.1.1] (((x v y)^ (z v y)) v u)^y=y.
135 [para_into,129.1.1,115.1.1] x v (y^x)=x.
136 [para_into,129.1.1,113.1.1,flip.1] (x v y)^y=y.
137 [para_from,130.1.1,112.1.1.2] ((x^y) v (y^z))^y= (x^y) v (y^z).
138 [para_into,133.1.1.1.1.2,125.1.1] (((x v y)^ (y v z)) v u)^y=y.
144 [para_into,136.1.1,123.1.1,flip.1] (x^y) v y=y.
147 [para_into,138.1.1.1,121.1.1] ((x v y)^ (y v z))^y=y.
159 [para_into,144.1.1.1,137.1.1] ((x^y) v (y^z)) v y=y.
161 [para_into,144.1.1.1,123.1.1] x v (y v x)=y v x.
162 [para_into,144.1.1.1,116.1.1] (x^y) v x=x.
170 [para_from,161.1.1,98.1.1.2] x^ (y v (x v z))=x.
173 [para_from,162.1.1,74.1.1.2] (((x v (y^ (z^u)))^ (y^ (z^u))) v (v^ (y^ (z^u))))^z=y^ (z^u).
178 [para_from,173.1.1,135.1.1.2] x v (y^ (x^z))=x.
181 [hyper,178,2,170,159,147] $ANS(step_all).

The proof exhibits variable richness 10, with but a single equation relying on ten distinct variables. I did not pursue a proof of strictly less variable richness. I offer you that bit of research, should you be looking for an intriguing topic.

As for some commentary of a historical nature, in Chapter 5 of the book titled *Automated Reasoning and the Discovery of Missing and Elegant Proofs* (by L. Wos and G. Pieper), you will find a comment about the lack of a proof of variable richness of 8 or less, in the context of McCune's first single axiom. What you have already read here answers that implied question in the affirmative. In Chapter 7 of the cited book, the following two challenges were offered.

CH08.LT: Does there exist a proof of length strictly less than 50 (applications of paramodulation) that deduces the given 4-basis from the first of the two 29-letter single axioms for lattice theory such that the proof relies solely on forward reasoning and does not rely on demodulation?

CH09.LT: Does there exist a short proof (say, of length 60 or less) with the second 29-letter single axiom as sole hypothesis that completes with the given 4-basis for lattice theory, where the proof is required to rely solely on forward reasoning and to avoid the use of demodulation?

The first of the two challenges has been met by the story told here. The second seems close to being met; indeed, I have a 65-step proof, which is not much longer than the sought-after 60-step proof. You might, perhaps based on what is offered here, meet the challenge as given.

Of course, I offer a mystery, at least for me. Specifically, those seven years ago, I tried very hard to obtain a proof shorter than length 50 for McCune's first single axiom. As reported here, I have succeeded (in 2009) in finding a 42-step proof. Precisely what is new I cannot, at the moment, discover. Obviously, I made some changes. Those years ago, I did apply an iterative approach, as I have narrated here. So, until I have some insight, I must remain mystified in the context of what modifications proved crucial, enabling OTTER to find the given 42-step proof when that proof clearly eluded both me and OTTER in 2002. If you intensely enjoy mysteries, you are implicitly offered a challenge, namely, to explain how in 2009 I was able to find what I could not find in 2002, a proof of length less than fifty.

3. Moufang Loops

Decades ago, Wayne Cowell, a colleague of mine at Argonne National Laboratory, told of an open question in an area of algebra focusing on what are called Moufang loops. In addition to the basic axioms, which I give shortly, any one of three identities (as he informed me), when added to those axioms, defines a Moufang loop. First, I list the basic axioms.

```
x = x.

x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)

%  identity:
1 * x = x .
x * 1 = x.

%  left cancellation
%  x*y != u  |  x*z != u  |  y = z.
%  right cancellation
%  y*x != u  |  z*x != u  |  y = z.
```

Then, when these axioms are augmented by any of the following three equations, one has a Moufang loop.

```
%  Axiom, Moufang 1:
(x * y) * (z * x)  = (x * (y * z)) * x.
%  Axiom, Moufang 2:
((x * y) * z) * y  = x * (y * (z * y)).
```

% Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.

Cowell told me that the three identities were the subject of an open question. Indeed, the three are provably equivalent, but, in a certain sense, four proofs were required to prove the equivalences, namely, 1 implies 2, 2 implies 3, 3 implies 2, and 2 implies 1. Ideally, three proofs should suffice. Put another way, could you find an ordering of the three Moufang identities such that, in the new ordering, 1a implies 2a without 3a participating, 2a implies 3a without 1a participating, and 3a implies 1a without 2a participating. Such a set of three proofs could be called, as I did name it, a circle of pure proofs.

You can think of the open question as asking whether a circle of (three) pure proofs exists for the three Moufang identities. By way of history, my first attempts those decades ago with a theorem-proving program we had produced failure. Only after many years did I succeed in finding the sought-after set of three pure proofs, with, of course, McCune's automated reasoning program OTTER. I produced a draft of an intended paper and sent it to the mathematician Ken Kunen. He informed me of a fourth identity, the following.

% Axiom, Moufang 4:
x * ((y * z) * x) = (x * y) * (z * x).

Upon receipt of his e-mail, I naturally sought a circle of four pure proofs, which I found with the method essentially unchanged. If my records are accurate, my first studies were conducted in 1995. Later, in 2002, I made more progress in refining the four pure proofs. You now have most of the important history; all that remains for this notebook is to present, where appropriate, the original proof I found (among the circle of pure proofs) and its input file—and, of course, the newer and shorter proof I recently found that improve upon the results obtained in 2002. (If you wish to understand the finer details about the input files I offer, a book that will prove useful is titled *A Fascinating Country in the World of Computing: Your Guide to Automated Reasoning;* my co-author is G. Pieper; the publisher is World Scientific.)

I note that the number of possible paths to pursue in search of a more elegant proof than that in hand is, with OTTER, monumental. My choices, from among that myriad, are based on experience and intuition and, yes, guesses. For but one example of the complexity, I have sometimes made progress by retreating to an earlier success, one less satisfying than that in hand, to enable the program to go down a different path. This approach to me is reminiscent of backing up on a road in order to take a different branch that might get you to your destination sooner in the long run. Indeed, when I sought a proof in group theory for McCune, who wished to have a 100-step proof, I reached a roadblock at length 255. If memory serves, I then returned to a 270-step proof coupled with choices of parameters and the like different from that which enabled me to find the 255-step proof, and I broke through. With that preamble in mind, I now turn to my studies, made here in the middle of 2009, of Moufang loops. The goal was, as noted, to find proofs more elegant in some manner than those I had in 2002.

I chose to seek a proof that Moufang 3 implies Moufang 4, while blocking the use of Mofang 1 and Moufang 2. I chose this question to study in part because the proof I had in 2002 has length 23, which means there might be room for improvement, perhaps substantial. As I report the journey, I must warn you that you will read about many phases. Along the way, I occasionally offer an appropriate input file and proof. For some, the methodology will be of greater interest than the actual treasure I found; for some, the treasure will be somewhat or very startling. As you will see, in the beginning, I had OTTER key on Moufang 3 by placing that equation in list(sos).

% Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.

In list(passive), I placed the following two forms of the negation of Moufang 4.

```
a* ((b*c)*a) !=  (a*b)* (c*a)|$ANS(m4).
(a*b)* (c*a) != a* ((b*c)*a) |$ANS(m4a).
```

In general, a good practice when trying to deduce some equation is to place two negations in list(passive, the second arising from interchanging the two arguments. You are thus protected from the program deducing one form of the target only. (I note, for the experienced researcher in automated reasoning, that, especially when seeking shorter proofs, I do not typically rely on a Knuth-Bendix approach; indeed, the presence in a proof of demodulation interferes with simple reporting of proof length.) With the express intention of modifying the input file that, in 2002, yielded a 23-step proof with the desired properties, I located the following input file.

An Original Input File for Studying Moufang 3

```
% Sample Input File for the Study of Moufang Loops
op(400,xfx, *). % make all association explicit
% set(knuth_bendix).

% set(knuth_bendix).
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
% set(ancestor_subsume).
set(back_sub).
% set(back_unit_deletion).
% assign(neg_weight,-5).
% set(print_proof_as_hints).

assign(max_weight,40).
% assign(change_limit_after,100).
% assign(new_max_weight,31).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
assign(pick_given_ratio,4).
% assign(max_seconds,30).

% The following list can be used to purge unwanted equations
weight_list(purge_gen).
% Blocking use of Moufang 1.
weight((((x * y) * (z * x) = (x * (y * z)) * x)),1000).
weight((((x * (y * z)) * x = ((x * y) * (z * x)))),1000).
% Blocking use of Moufang 2.
weight((((x * y) * z) * y = x * (y * (z * y))),1000).
weight(((x * (y * (z * y))) = (((x * y) * z) * y)),1000).
% % Blocking use of Moufang 3.
% weight((x * (y * (x * z)) = ((x * y) * x) * z), 1000).
% weight((((x * y) * x) * z = x * (y * (x * z))), 1000).
end_of_list.

% Used to complete applications of inference rules.
list(usable).
```

```
x = x.

x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)

%  identity:
1 * x = x .
x * 1 = x.

%  left cancellation
%  x*y != u  |  x*z != u  |  y = z.
%  right cancellation
%  y*x != u  |  z*x != u  |  y = z.
end_of_list.

%  Used to initiate applications of inference rules.
list(sos).
%  A consequence of left and right surjective:
%  It's all that is needed here.
%  x * R(x) = 1.
%  L(x) * x = 1.
%  The following negate left and right inverse.
%  d * y != 1.
%  y * e != 1.
%  actually, L and R turn out to be the same in a Moufang loop

%  Axiom, Moufang 1:
%  (x * y) * (z * x) = (x * (y * z)) * x.
%  Axiom, Moufang 2:
%  ((x * y) * z) * y = x * (y * (z * y)).
%  Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
%  Axiom, Moufang 4:
%  x * ((y * z) * x) = (x * y) * (z * x).
%  Negation Axiom, Moufang 1:
%  ((a * b) * (c * a)  != (a * (b * c)) * a)  |  $ANS(m1).
%  Negation Axiom, Moufang 2:
%  ((a * b) * c) * b  != a * (b * (c * b))  |  $ANS(m2).
%  Negation Axiom, Moufang 3:
%  a * (b * (a * c))  != ((a * b) * a) * c  |  $ANS(m3).
%  Negation Axiom, Moufang 4:
%  a * ((b * c) * a) != (a * b) * (c * a)  |  $ANS(m4).
end_of_list.

%  Used mainly to detect proof completion and to monitor progress.
list(passive).
a* ((b*c)*a) !=  (a*b)* (c*a)|$ANS(m4).
(a*b)* (c*a) != a* ((b*c)*a) |$ANS(m4a).
%  ((a * b) * (c * a)  != (a * (b * c)) * a)  |  $ANS(m1).
%  ((a * b) * c) * b  != a * (b * (c * b))  |  $ANS(m2).
%  a * (b * (a * c))  != ((a * b) * a) * c  |  $ANS(m3).
```

```
% a * ((b * c) * a) != (a * b) * (c * a)  |  $ANS(m4).
end_of_list.

%  The following list can be used to purge unwanted equations
list(demodulators).
%  Blocking use of Moufang 1.
%  EQ(((x * y) * (z * x) = (x * (y * z)) * x), $T).
%  EQ(((x * (y * z)) * x = ((x * y) * (z * x))), $T).
%  Blocking use of Moufang 2.
%  EQ((((x * y) * z) * y = x * (y * (z * y))), $T).
%  EQ(((x * (y * (z * y))) = (((x * y) * z) * y)), $T).
%  %  Blocking use of Moufang 3.
%  EQ((x * (y * (x * z)) = ((x * y) * x) * z), $T).
%  EQ((((x * y) * x) * z = x * (y * (x * z))), $T).
%  %  Blocking use of Moufang 4.
%   EQ((x * ((y * z) * x) = (x * y) * (z * x)), $T).
%   EQ(((x * y) * (z * x) = x * ((y * z) * x)), $T).
end_of_list.

list(hints).
x*rs(x,y)=y.
y=x*rs(x,y).
rs(x,x*y)=y.
y=rs(x,x*y).
ls(x,y)*y=x.
x=ls(x,y)*y.
ls(x*y,y)=x.
x=ls(x*y,y).
1*x=x.
x=1*x.
x*1=x.
x=x*1.
x=x.
x=x.
x* (y* (x*z))= ((x*y)*x)*z.
((x*y)*x)*z=x* (y* (x*z)).
a* ((b*c)*a)!= (a*b)* (c*a).
(a*b)* (c*a)!=a* ((b*c)*a).
((x*rs(x,y))*x)*z=x* (rs(x,y)* (x*z)).
x* (rs(x,y)* (x*z))= ((x*rs(x,y))*x)*z.
(x*y)*z=y* (rs(y,x)* (y*z)).
y* (rs(y,x)* (y*z))= (x*y)*z.
x* (rs(x,y)* (x*z))= (y*x)*z.
(y*x)*z=x* (rs(x,y)* (x*z)).
((x*y)*x)*1= (x*y)*x.
(x*y)*x= ((x*y)*x)*1.
((x*y)*x)*1=x* (y* (x*1)).
x* (y* (x*1))= ((x*y)*x)*1.
(x*y)*x=x* (y* (x*1)).
x* (y* (x*1))= (x*y)*x.
(x*y)*x=x* (y*x).
x* (y*x)= (x*y)*x.
x* (rs(x,y)* (x*rs(x,z)))= (y*x)*rs(x,z).
```

(y*x)*rs(x,z)=x* (rs(x,y)* (x*rs(x,z))).
x* (rs(x,y)*z)= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)*z).
(x*y)*rs(y,z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)*rs(y,z).
(1*x)*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)= (1*x)*rs(x,y).
x*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)=x*rs(x,y).
x=y* (rs(y,1)*x).
y* (rs(y,1)*x)=x.
x* (rs(x,1)*y)=y.
y=x* (rs(x,1)*y).
(ls(x,y)*y)*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)= (ls(x,y)*y)*rs(y,z).
x*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)=x*rs(y,z).
(x*rs(x,y))*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))*rs(rs(x,y),z).
x*rs(rs(y,x),z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x*rs(rs(y,x),z).
ls((x*y)*rs(y,z),rs(y,z))=x*y.
x*y=ls((x*y)*rs(y,z),rs(y,z)).
ls(x* (rs(x,y)*z),rs(x,z))=y*x.
y*x=ls(x* (rs(x,y)*z),rs(x,z)).
x* (rs(x,ls(y,x))*z)=y*rs(x,z).
y*rs(x,z)=x* (rs(x,ls(y,x))*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z).
y*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,ls(y,x))*1=rs(x,ls(y,x)).
rs(x,ls(y,x))=rs(x,ls(y,x))*1.
x* (rs(x,ls(y,x))*1)=y*rs(x,1).
y*rs(x,1)=x* (rs(x,ls(y,x))*1).
x*rs(x,ls(y,x))=y*rs(x,1).
y*rs(x,1)=x*rs(x,ls(y,x)).
x*rs(x,ls(y,x))=ls(y,x).
ls(y,x)=x*rs(x,ls(y,x)).
ls(x,y)=x*rs(y,1).
x*rs(y,1)=ls(x,y).
rs(rs(x,y),x)*1=rs(rs(x,y),x).
rs(rs(x,y),x)=rs(rs(x,y),x)*1.
rs(x,y)* (rs(rs(x,y),x)*1)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)* (rs(rs(x,y),x)*1).
rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)*rs(rs(x,y),x).
rs(x,y)*rs(rs(x,y),x)=x.
x=rs(x,y)*rs(rs(x,y),x).
x=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=x.
x*rs(rs(y,x),1)=y.
y=x*rs(rs(y,x),1).
ls(x* (rs(x,y)*z),rs(x,z))= (x* (rs(x,y)*z))*rs(rs(x,z),1).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=ls(x* (rs(x,y)*z),rs(x,z)).

(x* (rs(x,y)*z))*rs(rs(x,z),1)=y*x.
y*x= (x* (rs(x,y)*z))*rs(rs(x,z),1).
rs(x,y)*rs(rs(x,y),z)=z.
z=rs(x,y)*rs(rs(x,y),z).
(x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1)=y*x.
y*x= (x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
z*x= (x*y)*rs(rs(x,rs(rs(x,z),y)),1).
rs(x,x*rs(rs(y,x),1))=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,x*rs(rs(y,x),1)).
rs(x,y)=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,y).
rs(rs(x,y),1)=rs(y,x).
rs(y,x)=rs(rs(x,y),1).
rs(rs(x,rs(rs(x,y),z)),1)=rs(rs(rs(x,y),z),x).
rs(rs(rs(x,y),z),x)=rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(rs(x,z),y),x)=z*x.
z*x= (x*y)*rs(rs(rs(x,z),y),x).
(x*y)*rs(rs(rs(x,x*z),y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(rs(x,x*z),y),x).
(x*y)*rs(rs(z,y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(z,y),x).
(x*y)*rs(rs(z,y),x)=x* (z*x).
x* (z*x)= (x*y)*rs(rs(z,y),x).
x* (rs(x,y)* (x* (rs(x,1)*z)))= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)* (x* (rs(x,1)*z))).
x* (rs(x,y)*z)= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)*z).
(x*y)* (rs(y,1)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)* (rs(y,1)*z).
(x*rs(x,y))* (rs(rs(x,y),1)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))* (rs(rs(x,y),1)*z).
x* (rs(rs(y,x),1)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(rs(y,x),1)*z).
x* (rs(x,y)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(x,y)*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
x* (rs(x,y)*z)=x*rs(rs(y,x),z).
x*rs(rs(y,x),z)=x* (rs(x,y)*z).
rs(x,x*rs(rs(y,x),z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x*rs(rs(y,x),z)).
rs(x,x* (rs(x,y)*z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x* (rs(x,y)*z)).
rs(x,x* (rs(x,y)*z))=rs(x,y)*z.
rs(x,y)*z=rs(x,x* (rs(x,y)*z)).
rs(x,y)*z=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,y)*z.
rs(rs(x,y),z)=rs(y,x)*z.
rs(y,x)*z=rs(rs(x,y),z).
(x*y)* (rs(y,z)*x)=x* (z*x).
x* (z*x)= (x*y)* (rs(y,z)*x).
(x*y)* (rs(y,y*z)*x)=x* ((y*z)*x).

```
x* ((y*z)*x)= (x*y)* (rs(y,y*z)*x).
(x*y)* (z*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (z*x).
a* ((b*c)*a)= (a*b)* (c*a).
(a*b)* (c*a)=a* ((b*c)*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
$F.
end_of_list.

% Used for the hot list strategy.
list(hot).
% Axiom, Moufang 1:
% (x * y) * (z * x) = (x * (y * z)) * x.
% Axiom, Moufang 2:
% ((x * y) * z) * y = x * (y * (z * y)).
% Axiom, Moufang 3:
% x * (y * (x * z)) = ((x * y) * x) * z.
% Axiom, Moufang 4:
% x * ((y * z) * x) = (x * y) * (z * x).
x * rs(x,y) = y. % right solvable
rs(x, x * y) = y. % right solution is unique (implies left cancellation)
ls(x,y) * y = x. % left solvable
ls(x * y , y) = x. % left solution is unique (implies right cancellation)
% identity:
1 * x = x .
x * 1 = x.
end_of_list.
```

The given input file yields, as noted, a 23-step proof, the following (if all is in order). (As for the use of the *hot list strategy* and the occurrences of the term heat in a proof, I shall address those later when I focus on a study of deriving Moufang 3 from Moufang 2.)

A 23-Step Proof Deriving Moufang 4 from Moufang 3

```
----- Otter 3.2d, May 2002 -----
The process was started by wos on jaguar.mcs.anl.gov,
Thu Jun 6 14:14:09 2002
The command was "otter". The process ID is 1044.
----> UNIT CONFLICT at 30.72 sec ----> 44254 [binary,44253.1,10.1] $ANS(m4a).

Length of proof is 23. Level of proof is 12.

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

2 [] x*rs(x,y)=y.
4 [] ls(x,y)*y=x.
5 [] ls(x*y,y)=x.
6 [] 1*x=x.
7 [] x*1=x.
8 [] x* (y* (x*z))= ((x*y)*x)*z.
```

10 [] (a*b)* (c*a)!=a* ((b*c)*a)|$ANS(m4a).
161 [] x*rs(x,y)=y.
162 [] rs(x,x*y)=y.
166 [] x*1=x.
169 [para_into,8.1.1.2.2,7.1.1,flip.1] ((x*y)*x)*1=x* (y*x).
172 [para_into,8.1.1.2.2,2.1.1,flip.1] ((x*y)*x)*rs(x,z)=x* (y*z).
276 (heat=1) [para_into,169.1.1,166.1.1] (x*y)*x=x* (y*x).
352 (heat=1) [para_into,172.1.1.1.1,161.1.1] (x*y)*rs(y,z)=y* (rs(y,x)*z).
2562 [para_into,352.1.1.1,6.1.1,flip.1] x* (rs(x,1)*y)=x*rs(x,y).
2563 [para_into,352.1.1.1,4.1.1,flip.1] x* (rs(x,ls(y,x))*z)=y*rs(x,z).
2564 [para_into,352.1.1.1,2.1.1,flip.1] rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z).
2701 (heat=1) [para_into,2562.1.2,161.1.1] x* (rs(x,1)*y)=y.
2713 (heat=1) [para_into,2563.1.1.2,166.1.1] x*rs(x,ls(y,x))=y*rs(x,1).
2735 (heat=1) [para_into,2564.1.1.2,166.1.1] rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1).
3069 [para_from,352.1.1,5.1.1.1] ls(x* (rs(x,y)*z),rs(x,z))=y*x.
10300 [para_into,2713.1.1,2.1.1,flip.1] x*rs(y,1)=ls(x,y).
10639 [para_into,2735.1.1,2.1.1,flip.1] x*rs(rs(y,x),1)=y.
10728 (heat=1) [para_from,10639.1.1,162.1.1.2,flip.1] rs(rs(x,y),1)=rs(y,x).
16327 [para_into,10300.1.2,3069.1.1] (x* (rs(x,y)*z))*rs(rs(x,z),1)=y*x.
17439 [para_from,10728.1.1,2701.1.1.2.1] rs(x,y)* (rs(y,x)*z)=z.
17602 (heat=1) [para_into,17439.1.1.2,161.1.1,flip.1] rs(rs(x,y),z)=rs(y,x)*z.
20571 [para_into,16327.1.1.2,10728.1.1] (x* (rs(x,y)*z))*rs(z,x)=y*x.
21536 (heat=1) [para_into,20571.1.1.1.2,161.1.1] (x*y)*rs(rs(rs(x,z),y),x)=z*x.
30487 [para_into,21536.1.2,276.1.1] (x*y)*rs(rs(rs(x,x*z),y),x)=x* (z*x).
32605 (heat=1) [para_into,30487.1.1.2.1.1,162.1.1] (x*y)*rs(rs(z,y),x)=x* (z*x).
42670 [para_into,32605.1.1.2,17602.1.1] (x*y)* (rs(y,z)*x)=x* (z*x).
44253 (heat=1) [para_into,42670.1.1.2.1,162.1.1] (x*y)* (z*x)=x* ((y*z)*x).

Although not stated explicitly, the hot list was in use; indeed, the default is to assign the parameter heat the value 1. By way of a warning, you would be wise to examine the contents of the hot list carefully to avoid conducting an unwanted experiment. I have in mind, for example, the case in the context of proving Moufang 4 from Moufang 3 with, at the same time, avoiding the use of 1 and 2. If you accidentally have in the hot_list, say, 2, then it might be used in the proof, which would cost the result its purity. A good approach is to not rely on defaults where possible. In many of the experiments I am about to discuss, I assigned heat the value 1 when I was using the hot list strategy.

Specifically, I took the input file and made a number of changes. Certainly a crucial change was my invoking McCune's procedure called *ancestor subsumption*; indeed, I suspect that, without its use, the following events would not have occurred. Briefly, when ancestor subsumption is in use, the program compares paths to the same conclusion, preferring the strictly shorter one. Before discussing the other changes, I note as evidence of the value of using McCune's powerful procedure that the first experiment yielded two proofs, the first of length 23 (somewhat different from the 23-step proof found in 2002), and the second of length 21. If ancestor subsumption had not been present, then the second deduction of Moufang 4 would have been unnoticed and ignored. (Why I did not also experiment those years ago with ancestor subsumption I do not know.) Its deduction would have been discarded through the use of subsumption.

As for other changes, I assigned to pick_given_ratio the value 2 rather than 4. (With the assigned value of 2, the program chooses, for inference-rule initiation, 2 items by complexity, 1 by first come first serve, 2, 1, and the like.) With the new choice, more emphasis for drawing conclusions is placed on new conclusions retained early, regardless of their complexity. I assigned to max_distinct_vars the value 5, in contrast to no assignment in that regard, with the intention of curtailing the retention of new information, discarding any newly deduced item if it relied on six or more distinct variables. I assigned to max_weight the value 19 rather than 40 to enable OTTER to delve deeper into the search space, discarding any new item relying on twenty or more symbols. I explicitly assigned to heat the value 1. In addition to the many hints, whose origin I cannot now recall, I added a weight_list(pick_given) in which I placed twenty-three *resonators*, one corresponding to each of the original twenty-three deduced steps of the 2002 23-step proof. The value assigned to a resonator is almost always small and is used to direct a program's reasoning; the smaller the value, the higher the priority given to any item that matches a resonator. OTTER also offers a weight_list(purge_gen) to enable you to advise the program about what to discard upon generation, and the program offers you a weight_list(pick_and_purge) that combines directing and purging. The journey began encouragingly, witnessing a 21-step proof in place of the 23-step proof from 2002.

Iteration is my general approach. The key changes for the next useful experiment, occurring after some that yielded nothing, consisted of assigning the value 1 rather than 2 to the pick_given_ratio (to instruct the program to emphasize even more the use of early-retained items) and relying on the 21-step proof, using its steps as resonators. To make it easy to see what I did, I offer the following input file.

A Second Input File for Studying Moufang 3

```
% Sample Input File for the Study of Moufang Loops
op(400,xfx, *). % make all association explicit
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
set(ancestor_subsume).
set(back_sub).
assign(max_weight,19).
% assign(change_limit_after,100).
% assign(new_max_weight,10).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).
% set(sos_queue).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
assign(max_distinct_vars,5).
assign(pick_given_ratio,1).
assign(heat,1).
assign(max_seconds,25).


weight_list(pick_given).
% Following 21, with three not in the 23, prove the theorem
weight(x* (rs(x,y)* (x*z))= (y*x)*z,-2).
weight(x* (y* (x*1))= (x*y)*x,-2).
weight((x*y)*rs(y,z)=y* (rs(y,x)*z),-2).
weight((x*y)*x=x* (y*x),-2).
weight(x* (rs(x,1)*y)=x*rs(x,y),-2).
weight(rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z),-2).
weight((x*y)*rs(y,1)=y*rs(y,x),-2).
weight((x*y)*rs(y,rs(rs(y,x),z))=y*z,-2).
weight(x* (rs(x,1)*y)=y,-2).
weight(rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1),-2).
```

```
weight(rs(x,1)*y=rs(x,y),-2).
weight(x*rs(rs(y,x),1)=y,-2).
weight(rs(rs(x,y),1)=rs(y,x),-2).
weight(rs(rs(x,y),z)=rs(y,x)*z,-2).
weight((x*y)*rs(y,1)=x,-2).
weight((x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x,-2).
weight((x*y)*rs(rs(rs(x,z),y),x)=z*x,-2).
weight((x*y)*rs(rs(z,y),x)= (x*z)*x,-2).
weight((x*y)*rs(rs(z,y),x)=x* (z*x),-2).
weight((x*y)* (rs(y,z)*x)=x* (z*x),-2).
weight((x*y)* (z*x)=x* ((y*z)*x),-2).
end_of_list.

%  The following list can be used to purge unwanted equations
weight_list(purge_gen).
%  Blocking use of Moufang 1.
weight((((x * y) * (z * x) = (x * (y * z)) * x)),1000).
weight((((x * (y * z)) * x = ((x * y) * (z * x)))),1000).
%  Blocking use of Moufang 2.
weight((((x * y) * z) * y = x * (y * (z * y))),1000).
weight(((x * (y * (z * y))) = (((x * y) * z) * y)),1000).
end_of_list.

%  Used to complete applications of inference rules.
list(usable).
x = x.

x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)

%  identity:
1 * x = x .
x * 1 = x.

%  left cancellation
%  x*y != u  |  x*z != u  |  y = z.
%  right cancellation
%  y*x != u  |  z*x != u  |  y = z.
end_of_list.

%  Used to initiate applications of inference rules.
list(sos).
%  Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
end_of_list.

%  Used mainly to detect proof completion and to monitor progress.
list(passive).
a* ((b*c)*a) !=  (a*b)* (c*a)|$ANS(m4).
(a*b)* (c*a) != a* ((b*c)*a) |$ANS(m4a).
%  ((a * b) * (c * a)  != (a * (b * c)) * a)  |  $ANS(m1).
```

```
%  ((a * b) * c) * b  != a * (b * (c * b))  |  $ANS(m2).
%  a * (b * (a * c))  != ((a * b) * a) * c  |  $ANS(m3).
% a * ((b * c) * a) != (a * b) * (c * a)  |  $ANS(m4).
end_of_list.

%  The following list can be used to purge unwanted equations
list(demodulators).
%  Blocking use of Moufang 1.
%  EQ(((x * y) * (z * x) = (x * (y * z)) * x), $T).
%  EQ(((x * (y * z)) * x = ((x * y) * (z * x))), $T).
%  Blocking use of Moufang 2.
%  EQ((((x * y) * z) * y = x * (y * (z * y))), $T).
%  EQ(((x * (y * (z * y))) = (((x * y) * z) * y)), $T).
%  %  Blocking use of Moufang 3.
%  EQ((x * (y * (x * z)) = ((x * y) * x) * z), $T).
%  EQ((((x * y) * x) * z = x * (y * (x * z))), $T).
%  %  Blocking use of Moufang 4.
%   EQ((x * ((y * z) * x) = (x * y) * (z * x)), $T).
%   EQ(((x * y) * (z * x) = x * ((y * z) * x)), $T).
end_of_list.

list(hints).
x*rs(x,y)=y.
y=x*rs(x,y).
rs(x,x*y)=y.
y=rs(x,x*y).
ls(x,y)*y=x.
x=ls(x,y)*y.
ls(x*y,y)=x.
x=ls(x*y,y).
1*x=x.
x=1*x.
x*1=x.
x=x*1.
x=x.
x=x.
x* (y* (x*z))= ((x*y)*x)*z.
((x*y)*x)*z=x* (y* (x*z)).
a* ((b*c)*a)!= (a*b)* (c*a).
(a*b)* (c*a)!=a* ((b*c)*a).
((x*rs(x,y))*x)*z=x* (rs(x,y)* (x*z)).
x* (rs(x,y)* (x*z))= ((x*rs(x,y))*x)*z.
(x*y)*z=y* (rs(y,x)* (y*z)).
y* (rs(y,x)* (y*z))= (x*y)*z.
x* (rs(x,y)* (x*z))= (y*x)*z.
(y*x)*z=x* (rs(x,y)* (x*z)).
((x*y)*x)*1= (x*y)*x.
(x*y)*x= ((x*y)*x)*1.
((x*y)*x)*1=x* (y* (x*1)).
x* (y* (x*1))= ((x*y)*x)*1.
(x*y)*x=x* (y* (x*1)).
x* (y* (x*1))= (x*y)*x.
(x*y)*x=x* (y*x).
```

x* (y*x)= (x*y)*x.
x* (rs(x,y)* (x*rs(x,z)))= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)* (x*rs(x,z))).
x* (rs(x,y)*z)= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)*z).
(x*y)*rs(y,z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)*rs(y,z).
(1*x)*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)= (1*x)*rs(x,y).
x*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)=x*rs(x,y).
x=y* (rs(y,1)*x).
y* (rs(y,1)*x)=x.
x* (rs(x,1)*y)=y.
y=x* (rs(x,1)*y).
(ls(x,y)*y)*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)= (ls(x,y)*y)*rs(y,z).
x*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)=x*rs(y,z).
(x*rs(x,y))*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))*rs(rs(x,y),z).
x*rs(rs(y,x),z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x*rs(rs(y,x),z).
ls((x*y)*rs(y,z),rs(y,z))=x*y.
x*y=ls((x*y)*rs(y,z),rs(y,z)).
ls(x* (rs(x,y)*z),rs(x,z))=y*x.
y*x=ls(x* (rs(x,y)*z),rs(x,z)).
x* (rs(x,ls(y,x))*z)=y*rs(x,z).
y*rs(x,z)=x* (rs(x,ls(y,x))*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z).
y*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,ls(y,x))*1=rs(x,ls(y,x)).
rs(x,ls(y,x))=rs(x,ls(y,x))*1.
x* (rs(x,ls(y,x))*1)=y*rs(x,1).
y*rs(x,1)=x* (rs(x,ls(y,x))*1).
x*rs(x,ls(y,x))=y*rs(x,1).
y*rs(x,1)=x*rs(x,ls(y,x)).
x*rs(x,ls(y,x))=ls(y,x).
ls(y,x)=x*rs(x,ls(y,x)).
ls(x,y)=x*rs(y,1).
x*rs(y,1)=ls(x,y).
rs(rs(x,y),x)*1=rs(rs(x,y),x).
rs(rs(x,y),x)=rs(rs(x,y),x)*1.
rs(x,y)* (rs(rs(x,y),x)*1)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)* (rs(rs(x,y),x)*1).
rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)*rs(rs(x,y),x).
rs(x,y)*rs(rs(x,y),x)=x.
x=rs(x,y)*rs(rs(x,y),x).
x=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=x.
x*rs(rs(y,x),1)=y.
y=x*rs(rs(y,x),1).

ls(x* (rs(x,y)*z),rs(x,z))= (x* (rs(x,y)*z))*rs(rs(x,z),1).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=ls(x* (rs(x,y)*z),rs(x,z)).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=y*x.
y*x= (x* (rs(x,y)*z))*rs(rs(x,z),1).
rs(x,y)*rs(rs(x,y),z)=z.
z=rs(x,y)*rs(rs(x,y),z).
(x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1)=y*x.
y*x= (x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
z*x= (x*y)*rs(rs(x,rs(rs(x,z),y)),1).
rs(x,x*rs(rs(y,x),1))=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,x*rs(rs(y,x),1)).
rs(x,y)=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,y).
rs(rs(x,y),1)=rs(y,x).
rs(y,x)=rs(rs(x,y),1).
rs(rs(x,rs(rs(x,y),z)),1)=rs(rs(rs(x,y),z),x).
rs(rs(rs(x,y),z),x)=rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(rs(x,z),y),x)=z*x.
z*x= (x*y)*rs(rs(rs(x,z),y),x).
(x*y)*rs(rs(rs(x,x*z),y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(rs(x,x*z),y),x).
(x*y)*rs(rs(z,y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(z,y),x).
(x*y)*rs(rs(z,y),x)=x* (z*x).
x* (z*x)= (x*y)*rs(rs(z,y),x).
x* (rs(x,y)* (x* (rs(x,1)*z)))= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)* (x* (rs(x,1)*z))).
x* (rs(x,y)*z)= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)*z).
(x*y)* (rs(y,1)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)* (rs(y,1)*z).
(x*rs(x,y))* (rs(rs(x,y),1)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))* (rs(rs(x,y),1)*z).
x* (rs(rs(y,x),1)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(rs(y,x),1)*z).
x* (rs(x,y)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(x,y)*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
x* (rs(x,y)*z)=x*rs(rs(y,x),z).
x*rs(rs(y,x),z)=x* (rs(x,y)*z).
rs(x,x*rs(rs(y,x),z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x*rs(rs(y,x),z)).
rs(x,x* (rs(x,y)*z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x* (rs(x,y)*z)).
rs(x,x* (rs(x,y)*z))=rs(x,y)*z.
rs(x,y)*z=rs(x,x* (rs(x,y)*z)).
rs(x,y)*z=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,y)*z.
rs(rs(x,y),z)=rs(y,x)*z.
rs(y,x)*z=rs(rs(x,y),z).
(x*y)* (rs(y,z)*x)=x* (z*x).

```
x* (z*x)= (x*y)* (rs(y,z)*x).
(x*y)* (rs(y,y*z)*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (rs(y,y*z)*x).
(x*y)* (z*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (z*x).
a* ((b*c)*a)= (a*b)* (c*a).
(a*b)* (c*a)=a* ((b*c)*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
$F.
end_of_list.

%  Used for the hot list strategy.
list(hot).
%  Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)
%  identity:
1 * x = x .
x * 1 = x.
end_of_list.
```

The following 19-step proof shows the results obtained by using this input file.

A 19-Step Proof Deriving Moufang 4 from Moufang 3

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Fri Jul 17 17:27:29 2009
The command was "otter".  The process ID is 10376.

----> UNIT CONFLICT at   5.94 sec ----> 28707 [binary,28706.1,10.1] $ANS(m4a).

Length of proof is 19.  Level of proof is 11.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
6 [] 1*x=x.
7 [] x*1=x.
8 [] x* (y* (x*z))= ((x*y)*x)*z.
10 [] (a*b)* (c*a)!=a* ((b*c)*a)|$ANS(m4a).
161 [] x*rs(x,y)=y.
162 [] rs(x,x*y)=y.
166 [] x*1=x.
167 [para_into,8.1.1.2.2,7.1.1,flip.1] ((x*y)*x)*1=x* (y*x).
170 [para_into,8.1.1.2.2,2.1.1,flip.1] ((x*y)*x)*rs(x,z)=x* (y*z).
```

187 (heat=1) [para_into,167.1.1,166.1.1] (x*y)*x=x* (y*x).
231 (heat=1) [para_into,170.1.1.1.1,161.1.1] (x*y)*rs(y,z)=y* (rs(y,x)*z).
4244 [para_into,231.1.1.1,6.1.1,flip.1] x* (rs(x,1)*y)=x*rs(x,y).
4250 [para_into,231.1.2.2,7.1.1] (x*y)*rs(y,1)=y*rs(y,x).
4251 [para_into,231.1.2.2,2.1.1] (x*y)*rs(y,rs(rs(y,x),z))=y*z.
4269 (heat=1) [para_into,4244.1.2,161.1.1] x* (rs(x,1)*y)=y.
4325 (heat=1) [para_into,4250.1.2,161.1.1] (x*y)*rs(y,1)=x.
5494 [para_into,4325.1.1.1,4251.1.1] (x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
5525 [para_into,4325.1.1.1,2.1.1] x*rs(rs(y,x),1)=y.
5831 (heat=1) [para_from,5525.1.1,162.1.1.2,flip.1] rs(rs(x,y),1)=rs(y,x).
10311 [para_from,4269.1.1,3.1.1.2,flip.1] rs(x,1)*y=rs(x,y).
19387 [para_from,5831.1.1,5494.1.1.2] (x*y)*rs(rs(rs(x,z),y),x)=z*x.
19402 [para_from,5831.1.1,10311.1.1.1,flip.1] rs(rs(x,y),z)=rs(y,x)*z.
24727 [para_into,19387.1.2,187.1.1] (x*y)*rs(rs(rs(x,x*z),y),x)=x* (z*x).
25104 (heat=1) [para_into,24727.1.1.2.1.1,162.1.1] (x*y)*rs(rs(z,y),x)=x* (z*x).
28476 [para_into,25104.1.1.2,19402.1.1] (x*y)* (rs(y,z)*x)=x* (z*x).
28706 (heat=1) [para_into,28476.1.1.2.1,162.1.1] (x*y)* (z*x)=x* ((y*z)*x).

I viewed this proof as quite satisfying. Some gold is being mined! Perhaps the mine would yield more.

Various experiments yielded nothing, however, so after a while I turned to *cramming*. Recall that with (at least one incarnation of) cramming, you seek the proofs of the various deduced steps of a specific proof, in this case a 19-step proof. For the record, say in the case of proving a conjunction where you have chosen a subproof of one of the members of the conjunction to be proved, the concern is not for the lengths of the proofs of the individual members, but rather for the length of the proof of the entire conjunction. In fact, with successful cramming, often the new proofs of the individual members are longer than those found in the proof (of the conjunction) in focus. I chose a 13-step proof of step 17 of the 19-step proof in hand, placing those equations in list(sos) with the goal of finding a proof of length 5 or less that deduces Moufang 4, whether its arguments are interchanged or not. I switched my approach, as is so typical with cramming, from an approach relying on the ratio strategy, which keys in part on the complexity of items, to an approach that keys on first come first serve, namely, a level-saturation approach (or breadth-first search). To do so, I commented out pick_given_ratio and commented in sos_queue. And again an unexplained occurrence was encountered; indeed, the pick_gen weight_list contained, in this third experiment, twenty-one resonators, corresponding to the 21-step proof found earlier. Why the 19-step was not keyed on I cannot explain. I did assign the value 1 to heat.

Of course, cramming is not a panacea. On the other hand, it must have succeeded, or you would not be reading these words. It did, yielding a 5-step proof—after more than 1,100,000 new equations had been retained and after almost 24,000 CPU-seconds had been expended. In other words, there existed an exceedingly strong suggestion that the use of the thirteen steps crammed on together with the appropriate five deduced steps would yield a new 18-step proof.

Being aware that a proof based on complexity supplied by a level-saturation approach can contain unexpected steps—for example, complex steps in that such an approach is forced to consider new conclusions in order—I wondered what the new proof would be like. The 18-step proof that was promised was obtained by placing resonators corresponding to the 13 formulas crammed on, followed by the five that were deduced.

An 18-Step Proof

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Wed Jul 8 21:26:01 2009
The command was "otter". The process ID is 7233.

----> UNIT CONFLICT at 0.03 sec ----> 532 [binary,531.1,17.1] $ANS(m4a).

Length of proof is 18. Level of proof is 12.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
6 [] 1*x=x.
7 [] x*1=x.
8 [] x* (y* (x*z))= ((x*y)*x)*z.
17 [] (a*b)* (c*a)!=a* ((b*c)*a)|$ANS(m4a).
29 [para_into,8.1.1.2.2,7.1.1,flip.1] ((x*y)*x)*1=x* (y*x).
32 [para_into,8.1.1.2.2,2.1.1,flip.1] ((x*y)*x)*rs(x,z)=x* (y*z).
53 [para_into,29.1.1,7.1.1] (x*y)*x=x* (y*x).
68 [para_into,32.1.1.1.1,2.1.1] (x*y)*rs(y,z)=y* (rs(y,x)*z).
125 [para_into,68.1.2.2,7.1.1] (x*y)*rs(y,1)=y*rs(y,x).
126 [para_into,68.1.2.2,2.1.1] (x*y)*rs(y,rs(rs(y,x),z))=y*z.
148 [para_into,125.1.2,2.1.1] (x*y)*rs(y,1)=x.
158 [para_into,126.1.1.1,6.1.1] x*rs(x,rs(rs(x,1),y))=x*y.
177 [para_into,148.1.1.1,126.1.1] (x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
192 [para_into,148.1.1.1,2.1.1] x*rs(rs(y,x),1)=y.
211 [para_into,158.1.1,2.1.1] rs(rs(x,1),y)=x*y.
310 [para_from,192.1.1,3.1.1.2,flip.1] rs(rs(x,y),1)=rs(y,x).
377 [para_from,310.1.1,177.1.1.2] (x*y)*rs(rs(rs(x,z),y),x)=z*x.
421 [para_into,377.1.2,53.1.1] (x*y)*rs(rs(rs(x,x*z),y),x)=x* (z*x).
440 [para_into,421.1.1.2.1.1,3.1.1] (x*y)*rs(rs(z,y),x)=x* (z*x).
460 [para_into,440.1.1.2.1,310.1.2] (x*y)*rs(rs(rs(y,z),1),x)=x* (z*x).
502 [para_into,460.1.1.2,211.1.1] (x*y)* (rs(y,z)*x)=x* (z*x).
531 [para_into,502.1.1.2.1,3.1.1] (x*y)* (z*x)=x* ((y*z)*x).

The 18-step, level-12 proof contains three steps not present in the 19-step proof.

At this point, I must admit I thought it likely that I would find no shorter proof than length 18. Still, a few additional experiments seemed in order. (By the way, when I talk about a third experiment, as indicated, I am taking poetic license in that many are not being reported here, just the highlights.) So, in emulation of that which had worked, I returned to cramming. In particular, I took a 12-step proof of what I think was step 14 of the given 19-step proof and inserted those equations in list(sos), again with the idea of forcing those equations to be crammed into (I hoped) a new proof meriting study. I again assigned the value 19 to max_weight. However, because I thought OTTER might be forced to process many items to initiate inference-rule application, I added the following two commands.

assign(change_limit_after,200).
assign(new_max_weight,15).

The first of the two commands instructs the program to change the assigned value of max_weight after 200 items have been chosen and used to initiate inference-rule application. The second of the two commands instructs the program to assign the value 15 to max_weight. In other words, with the addition of these two commands, the program will delve deeper into the search space than it would if the max_weight had remained with an assignment of the value 19. I assigned the value 1 to heat, a move that brings up an important point, especially in the context of a depth-first (level-saturation) approach. Specifically, when a new conclusion is retained, before it is considered for being the focus of attention in the context of those already deduced and retained, the new one is considered with all of the members of the (initial) hot list). With heat=1, an examination of the output of a level-saturation run will show that the program adjoins clauses at level greater than *k* when it is, supposedly, just considering level *k*. I used as resonators the equations from the 19-step proof. In just a bit over a half an hour of computing time, with the retention of a clause numbering just over 280,000, a 3-step proof was found. Of course, I was elated, for I thought I now had the makings of a 15-step proof. Such was not the case.

In fact, if memory serves, when I tried to return to an approach based on McCune's ratio strategy, relying on the fifteen resonators, twelve from those corresponding to the twelve on which I crammed and three that were deduced, OTTER presented me with a new 19-step proof. I was surprised and, yes, possibly annoyed. No, I did not take the time to find out what caused the unwanted (for the moment) discovery. Instead, I began blocking various steps of the new 19-step proof, instructing OTTER to immediately discard certain of them when deduced. Usually, I use demodulation to achieve such an objective, demodulating unwanted items to junk. For some reason, however, I had to turn to a weight_list(purge_gen), eventually adjoining four equations. I now present the purge_gen list I used, and then I give the somewhat unexpected results.

```
weight_list(purge_gen).
weight(rs(x,y)*rs(rs(x,y),rs(rs(y,x),z))=rs(x,y)*z,1000).
weight(x* (rs(x,y)* (x*z))= (y*x)*z,1000).
weight(x* (y* (x*1))= (x*y)*x,1000).
weight(x* (rs(x,1)*y)=x*rs(x,y),1000).
% Blocking use of Moufang 1.
weight((((x * y) * (z * x) = (x * (y * z)) * x)),1000).
weight((((x * (y * z)) * x = ((x * y) * (z * x)))),1000).
% Blocking use of Moufang 2.
weight((((x * y) * z) * y = x * (y * (z * y))),1000).
weight(((x * (y * (z * y))) = (((x * y) * z) * y)),1000).
% % Blocking use of Moufang 3.
% weight((x * (y * (x * z)) = ((x * y) * x) * z), 1000).
% weight((((x * y) * x) * z = x * (y * (x * z))), 1000).
end_of_list.
```

The given purge_gen list was used, as noted, in an experiment that occurred after I found a supposed 3-step proof that would complete a proof of Moufang 4, relying on cramming with twelve steps from a 19-step proof. In this key experiment, which I now think of as the fifth of the important ones, I dispensed with the use of that lengthy list of hints, used 19 as the value assigned to max_weight, used the value 1 assigned to heat, and used fifteen resonators (already described). To my annoyance, OTTER completed an 18-step proof, the following.

A Later 18-Step Proof

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on octopus.mcs.anl.gov,
Sun Jul 12 11:04:26 2009
The command was "otter". The process ID is 14013.

----> UNIT CONFLICT at   0.81 sec ----> 12456 [binary,12455.1,29.1] $ANS(m4a).

Length of proof is 18. Level of proof is 10.

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

2 [] x*rs(x,y)=y.
7 [] x*1=x.
8 [] x* (y* (x*z))= ((x*y)*x)*z.
29 [] (a*b)* (c*a)!=a* ((b*c)*a)|$ANS(m4a).
31 [] x*rs(x,y)=y.
```

```
32 [] rs(x,x*y)=y.
35 [] 1*x=x.
36 [] x*1=x.
37 [para_into,8.1.1.2.2,7.1.1,flip.1] ((x*y)*x)*1=x* (y*x).
41 [para_into,8.1.1.2.2,2.1.1,flip.1] ((x*y)*x)*rs(x,z)=x* (y*z).
57 (heat=1) [para_into,37.1.1,36.1.1] (x*y)*x=x* (y*x).
102 (heat=1) [para_into,41.1.1.1.1,31.1.1] (x*y)*rs(y,z)=y* (rs(y,x)*z).
852 [para_into,102.1.2.2,7.1.1] (x*y)*rs(y,1)=y*rs(y,x).
854 [para_into,102.1.2.2,2.1.1] (x*y)*rs(y,rs(rs(y,x),z))=y*z.
956 (heat=1) [para_into,852.1.2,31.1.1] (x*y)*rs(y,1)=x.
962 (heat=1) [para_into,854.1.1.1,35.1.1] x*rs(x,rs(rs(x,1),y))=x*y.
2454 [para_into,956.1.1.1,854.1.1] (x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
2475 [para_into,956.1.1.1,2.1.1] x*rs(rs(y,x),1)=y.
2697 (heat=1) [para_from,2475.1.1,32.1.1.2,flip.1] rs(rs(x,y),1)=rs(y,x).
3331 [para_into,962.1.1,2.1.1] rs(rs(x,1),y)=x*y.
5530 [para_from,2697.1.1,2454.1.1.2] (x*y)*rs(rs(rs(x,z),y),x)=z*x.
6142 [para_into,3331.1.1.1,2697.1.1] rs(rs(x,y),z)=rs(y,x)*z.
6191 (heat=1) [para_into,6142.1.2.1,32.1.1] rs(rs(x*y,x),z)=y*z.
8241 [para_into,5530.1.2,57.1.1] (x*y)*rs(rs(rs(x,x*z),y),x)=x* (z*x).
8474 (heat=1) [para_into,8241.1.1.2.1.1,32.1.1] (x*y)*rs(rs(z,y),x)=x* (z*x).
12455 [para_into,8474.1.1.2,6191.1.1] (x*y)* (z*x)=x* ((y*z)*x).
```

My annoyance quickly abated, however, when I noticed that this proof has level 10, not 12 as is true of the earlier 18-step proof. By applying another of McCune's program options, for set-theoretic difference, I found that two of the proof steps were not present in the 18-step, level-12 proof. Therefore, hope was rekindled. Indeed, perhaps I could use this new 18-step proof to get farther, to find more gold. (You did notice, I am sure, that no guarantee exists when taking the results of a cramming run, of finding a shorter proof than that in hand, or, for that matter, of completing the expected proof.)

With a small change, namely, commenting out two of the four demodulators, I submitted another experiment, and the program found two proofs, the first of length 19 and the second of length 18 and level 9; a comparison showed that the latter avoids five of the step of the 18-step, level-10 proof. To present that proof to me, the program retained almost one million new equations, requiring approximately 2.25 CPU-hours. That 18-step proof of level 9 was immediately put to use, as the following input file shows.

A Third Input File for the Study of Moufang 3

```
%  Sample Input File for the Study of Moufang Loops
op(400,xfx, *).  %  make all association explicit
%  set(knuth_bendix).

%  set(knuth_bendix).
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
set(ancestor_subsume).
set(back_sub).
%  set(back_unit_deletion).
%  assign(neg_weight,-5).
%  set(print_proof_as_hints).

assign(max_weight,19).
assign(change_limit_after,200).
```

```
assign(new_max_weight,15).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).
% set(sos_queue).
% set(process_input).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
assign(max_distinct_vars,4).
assign(pick_given_ratio,1).
assign(heat,1).
assign(max_seconds,2).


weight_list(pick_given).
% Following 18/9, smallest size as of 07-12-09
weight(((x*y)*x)*1=x* (y*x),2).
weight(((x*y)*x)*rs(x,z)=x* (y*z),2).
weight((x*y)*x=x* (y*x),2).
weight((x*y)*rs(y,z)=y* (rs(y,x)*z),2).
weight((x* (y*x))*rs(x,z)=x* (y*z),2).
weight((x* (y*x))*rs(x,rs(y,z))=x*z,2).
weight(x* (rs(x,1)*y)=x*rs(x,y),2).
weight((x*y)*rs(y,1)=y*rs(y,x),2).
weight(x*rs(x,rs(rs(x,1),y))=x*y,2).
weight((x*y)*rs(y,1)=x,2).
weight(rs(rs(x,1),y)=x*y,2).
weight(x*rs(rs(y,x),1)=y,2).
weight(rs(rs(x,y),1)=rs(y,x),2).
weight(rs(x*y,x)=rs(y,1),2).
weight((x*rs(y,z))*rs(z,y)=x,2).
weight(rs(rs(x*y,x),z)=y*z,2).
weight((x*y)*rs(rs(z,y),x)=x* (z*x),2).
weight((x*y)* (z*x)=x* ((y*z)*x),2).
end_of_list.

% The following list can be used to purge unwanted equations
weight_list(purge_gen).
% Blocking use of Moufang 1.
weight((((x * y) * (z * x) = (x * (y * z)) * x)),1000).
weight((((x * (y * z)) * x = ((x * y) * (z * x)))),1000).
% Blocking use of Moufang 2.
weight((((x * y) * z) * y = x * (y * (z * y))),1000).
weight(((x * (y * (z * y))) = (((x * y) * z) * y)),1000).
weight(junk,1000).
end_of_list.

% Used to complete applications of inference rules.
list(usable).
```

```
x = x.

x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)

%  identity:
1 * x = x .
x * 1 = x.

%  left cancellation
%  x*y != u  |  x*z != u  |  y = z.
%  right cancellation
%  y*x != u  |  z*x != u  |  y = z.
end_of_list.

%  Used to initiate applications of inference rules.
list(sos).
%  Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
end_of_list.

%  Used mainly to detect proof completion and to monitor progress.
list(passive).
%  Following  19 negs for a 19-step proof that 3 imp 4 pure with 1and2.
((a*b)*a)*1!=a* (b*a) | $ANS(INTER).
((a*b)*a)*rs(a,c)!=a* (b*c) | $ANS(INTER).
(a*b)*a!=a* (b*a) | $ANS(INTER).
(a*b)*rs(b,c)!=b* (rs(b,a)*c) | $ANS(INTER).
a* (rs(a,1)*b)!=a*rs(a,b) | $ANS(INTER).
(a*b)*rs(b,1)!=b*rs(b,a) | $ANS(INTER).
(a*b)*rs(b,rs(rs(b,a),c))!=b*c | $ANS(INTER).
a* (rs(a,1)*b)!=b | $ANS(INTER).
(a*b)*rs(b,1)!=a | $ANS(INTER).
(a*b)*rs(rs(a,rs(rs(a,c),b)),1)!=c*a | $ANS(INTER).
a*rs(rs(b,a),1)!=b | $ANS(INTER).
rs(rs(a,b),1)!=rs(b,a) | $ANS(INTER).
rs(a,1)*b!=rs(a,b) | $ANS(INTER).
(a*b)*rs(rs(rs(a,c),b),a)!=c*a | $ANS(INTER).
rs(rs(a,b),c)!=rs(b,a)*c | $ANS(INTER).
(a*b)*rs(rs(rs(a,a*c),b),a)!=a* (c*a) | $ANS(INTER).
(a*b)*rs(rs(c,b),a)!=a* (c*a) | $ANS(INTER).
(a*b)* (rs(b,c)*a)!=a* (c*a) | $ANS(INTER).
(a*b)* (c*a)!=a* ((b*c)*a) | $ANS(INTER).
a* ((b*c)*a) !=  (a*b)* (c*a)|$ANS(m4).
(a*b)* (c*a) != a* ((b*c)*a) |$ANS(m4a).
%  ((a * b) * (c * a)  != (a * (b * c)) * a)  |  $ANS(m1).
%  ((a * b) * c) * b  != a * (b * (c * b))  |  $ANS(m2).
%  a * (b * (a * c))  != ((a * b) * a) * c  |  $ANS(m3).
% a * ((b * c) * a) != (a * b) * (c * a)  |  $ANS(m4).
end_of_list.
```

```
% The following list can be used to purge unwanted equations
list(demodulators).
EQ(x*rs(rs(y,x),1)=y,junk).
end_of_list.

list(hints).
end_of_list.

% Used for the hot list strategy.
list(hot).
% Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
x * rs(x,y) = y.  % right solvable
rs(x, x * y) = y.  % right solution is unique (implies left cancellation)
ls(x,y) * y = x.  % left solvable
ls(x * y , y) = x.  % left solution is unique (implies right cancellation)
% identity:
1 * x = x .
x * 1 = x.
end_of_list.
```

The following was obtained with this third input file.

A 17-Step Proof

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Sat Jul 18 18:04:57 2009
The command was "otter".  The process ID is 8014.

----> UNIT CONFLICT at   0.63 sec ----> 11976 [binary,11975.1,29.1] $ANS(m4a).

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

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
6 [] 1*x=x.
7 [] x*1=x.
8 [] x* (y* (x*z))= ((x*y)*x)*z.
29 [] (a*b)* (c*a)!=a* ((b*c)*a)|$ANS(m4a).
32 [] x*rs(x,y)=y.
37 [] x*1=x.
38 [para_into,8.1.1.2.2,7.1.1,flip.1] ((x*y)*x)*1=x* (y*x).
42 [para_into,8.1.1.2.2,2.1.1,flip.1] ((x*y)*x)*rs(x,z)=x* (y*z).
60 (heat=1) [para_into,38.1.1,37.1.1] (x*y)*x=x* (y*x).
105 (heat=1) [para_into,42.1.1.1.1,32.1.1] (x*y)*rs(y,z)=y* (rs(y,x)*z).
628 [para_from,60.1.1,42.1.1.1] (x* (y*x))*rs(x,z)=x* (y*z).
706 (heat=1) [para_into,628.1.2.2,32.1.1] (x* (y*x))*rs(x,rs(y,z))=x*z.
859 [para_into,105.1.1.1,6.1.1,flip.1] x* (rs(x,1)*y)=x*rs(x,y).
867 [para_into,105.1.2.2,7.1.1] (x*y)*rs(y,1)=y*rs(y,x).
898 (heat=1) [para_into,859.1.1.2,32.1.1,flip.1] x*rs(x,rs(rs(x,1),y))=x*y.
```

```
975 (heat=1) [para_into,867.1.2,32.1.1] (x*y)*rs(y,1)=x.
3560 [para_into,898.1.1,2.1.1] rs(rs(x,1),y)=x*y.
4399 [para_from,975.1.1,3.1.1.2] rs(x*y,x)=rs(y,1).
7586 [para_from,4399.1.2,975.1.1.2] (x*y)*rs(z*y,z)=x.
7590 [para_from,4399.1.2,3560.1.1.1] rs(rs(x*y,x),z)=y*z.
7691 (heat=1) [para_into,7586.1.1.2.1,32.1.1] (x*rs(y,z))*rs(z,y)=x.
11057 [para_into,7691.1.1.1,706.1.1] (x*y)*rs(rs(z,y),x)=x* (z*x).
11975 [para_into,11057.1.1.2,7590.1.1] (x*y)* (z*x)=x* ((y*z)*x).
```

At this point, I was indeed pleased but, as expected, wondered whether a bit more was obtainable.

Well, if cramming worked before, why not try it again? So, I returned to level saturation, adjoined to list(sos) twelve equations, from a 12-step proof of step 16 of the cited 17-step proof, and made what might be accurately termed an odd move. In particular, in the weight_list(pick_given), I placed as resonators those twenty-one used earlier, those corresponding to the 21-step proof I obtained at the beginning of this journey. After almost 14,000 CPU-seconds and the retention of more than one million new conclusions, OTTER found a 5-step proof of Moufang 4. Now, a reasonable conjecture would be that no progress had occurred. However, a comparison of the five new steps with the 17-step proof in hand showed that four of the five were not in that 17-step proof. In other words, a new 17-step proof may be near, and that proof (if it exists) might lead somewhere. What might you do at this point? To enable you to participate, I shall make a few short remarks before supplying the input file I used and the results I obtained.

A glance at the input files and proofs I give here shows that the inclusion of the negation of Moufang 4 in two forms proved crucial. Of course, as noted, I mean that I included, so-to-speak, a != b and b != a, where a and b are, respectively, the negation of the two arguments of Moufang 4. Also meriting comment is the inclusion, in list(usable), of $x = x$, which might seem a bit puzzling. The explanation rests with the following. When paramodulation is in use, directly or through a Knuth-Bendix approach, a program might deduce, for some expression c, c != c. Such a deduction should be treated as the completion of a proof, all things being equal.

And I now present the input file I used to seek additional progress.

A Fourth Input File

```
% Sample Input File for the Study of Moufang Loops
op(400,xfx, *). % make all association explicit
% set(knuth_bendix).

% set(knuth_bendix).
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
set(ancestor_subsume).
set(back_sub).
% set(back_unit_deletion).
% assign(neg_weight,-5).
% set(print_proof_as_hints).

assign(max_weight,19).
assign(change_limit_after,200).
assign(new_max_weight,15).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).
% set(sos_queue).
```

```
% set(process_input).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
assign(max_distinct_vars,4).
assign(pick_given_ratio,1).
assign(heat,1).
assign(max_seconds,2).


weight_list(pick_given).
% Following 12 prove step 16 of a 17step proof that 3 imp 4, pure with 1and2.
weight(((x*y)*x)*1=x* (y*x),1).
weight(((x*y)*x)*rs(x,z)=x* (y*z),1).
weight((x*y)*x=x* (y*x),1).
weight((x*y)*rs(y,z)=y* (rs(y,x)*z),1).
weight((x* (y*x))*rs(x,z)=x* (y*z),1).
weight((x* (y*x))*rs(x,rs(y,z))=x*z,1).
weight((x*y)*rs(y,1)=y*rs(y,x),1).
weight((x*y)*rs(y,1)=x,1).
weight(rs(x*y,x)=rs(y,1),1).
weight((x*y)*rs(z*y,z)=x,1).
weight((x*rs(y,z))*rs(z,y)=x,1).
weight((x*y)*rs(rs(z,y),x)=x* (z*x),1).
% Following 5 from cramming on preceding 12, temp.moufang.3imp4.new.out1y8.
weight((x* (y*x))*rs(x,rs(z*y,z))=x*1,1).
weight((x* (y*x))*rs(x,rs(z*y,z))=x,1).
weight((x*y)*rs(z*rs(x,rs(u,y)),z)=x* (u*x),1).
weight((x*y)*rs(x,x* (z*x))=x* ((y*z)*x),1).
weight((x*y)* (z*x)=x* ((y*z)*x),1).
end_of_list.

% The following list can be used to purge unwanted equations of various types.
weight_list(purge_gen).
%weight(rs(x,y)*rs(rs(x,y),rs(rs(y,x),z))=rs(x,y)*z,1000).
%weight(x* (rs(x,y)* (x*z))= (y*x)*z,1000).
%weight(x* (y* (x*1))= (x*y)*x,1000).
%weight(x* (rs(x,1)*y)=x*rs(x,y),1000).
% Blocking use of Moufang 1.
weight((((x * y) * (z * x) = (x * (y * z)) * x)),1000).
weight((((x * (y * z)) * x = ((x * y) * (z * x)))),1000).
% Blocking use of Moufang 2.
weight(((((x * y) * z) * y = x * (y * (z * y))),1000).
weight(((x * (y * (z * y))) = (((x * y) * z) * y)),1000).
weight(junk,1000).
end_of_list.

% Used to complete applications of inference rules.
list(usable).
```

```
x = x.

x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)

%  identity:
1 * x = x .
x * 1 = x.

%  left cancellation
%  x*y != u  |  x*z != u  |  y = z.
%  right cancellation
%  y*x != u  |  z*x != u  |  y = z.
end_of_list.

%  Used to initiate applications of inference rules.
list(sos).
%  Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
end_of_list.

%  Used mainly to detect proof completion and to monitor progress.
list(passive).
a* ((b*c)*a) !=  (a*b)* (c*a)|$ANS(m4).
(a*b)* (c*a) != a* ((b*c)*a) |$ANS(m4a).
end_of_list.

%  The following list can be used to purge unwanted equations of various types.
list(demodulators).
%  EQ(x*rs(rs(y,x),1)=y,junk).
%  Blocking use of Moufang 1.
%  EQ(((x * y) * (z * x) = (x * (y * z)) * x), $T).
%  EQ(((x * (y * z)) * x = ((x * y) * (z * x))), $T).
%  Blocking use of Moufang 2.
%  EQ((((x * y) * z) * y = x * (y * (z * y))), $T).
%  EQ(((x * (y * (z * y))) = (((x * y) * z) * y)), $T).
%  %  Blocking use of Moufang 3.
%  EQ((x * (y * (x * z)) = ((x * y) * x) * z), $T).
%  EQ((((x * y) * x) * z = x * (y * (x * z))), $T).
%  %  Blocking use of Moufang 4.
%   EQ((x * ((y * z) * x) = (x * y) * (z * x)), $T).
%   EQ(((x * y) * (z * x) = x * ((y * z) * x)), $T).
end_of_list.

list(hints).
end_of_list.

%  Used for the hot list strategy.
list(hot).
%  Axiom, Moufang 3:
x * (y * (x * z)) = ((x * y) * x) * z.
```

```
x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)
%  identity:
1 * x = x .
x * 1 = x.
end_of_list.
```

A glance at the file reveals my use of the seventeen resonators in focus a bit ago, the twelve I crammed on and the five that were deduced from that cramming.

When OTTER focused on that input file, the following 15-step proof resulted—amazing to me.

A Startling 15-Step Proof

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Sun Jul 19 16:36:15 2009
The command was "otter".  The process ID is 2264.

----> UNIT CONFLICT at   0.39 sec ----> 8577 [binary,8576.1,10.1] $ANS(m4a).

Length of proof is 15.  Level of proof is 9.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
7 [] x*1=x.
8 [] x* (y* (x*z))= ((x*y)*x)*z.
10 [] (a*b)* (c*a)!=a* ((b*c)*a)|$ANS(m4a).
12 [] x*rs(x,y)=y.
13 [] rs(x,x*y)=y.
17 [] x*1=x.
18 [para_into,8.1.1.2.2,7.1.1,flip.1] ((x*y)*x)*1=x* (y*x).
21 [para_into,8.1.1.2.2,2.1.1,flip.1] ((x*y)*x)*rs(x,z)=x* (y*z).
38 (heat=1) [para_into,18.1.1,17.1.1] (x*y)*x=x* (y*x).
82 (heat=1) [para_into,21.1.1.1.1,12.1.1] (x*y)*rs(y,z)=y* (rs(y,x)*z).
604 [para_from,38.1.1,21.1.1.1] (x* (y*x))*rs(x,z)=x* (y*z).
682 (heat=1) [para_into,604.1.2.2,12.1.1] (x* (y*x))*rs(x,rs(y,z))=x*z.
842 [para_into,82.1.2.2,7.1.1] (x*y)*rs(y,1)=y*rs(y,x).
947 (heat=1) [para_into,842.1.2,12.1.1] (x*y)*rs(y,1)=x.
3519 [para_from,947.1.1,3.1.1.2] rs(x*y,x)=rs(y,1).
5453 [para_from,3519.1.2,947.1.1.2] (x*y)*rs(z*y,z)=x.
5457 [para_from,3519.1.2,682.1.1.2.2] (x* (y*x))*rs(x,rs(z*y,z))=x*1.
5559 (heat=1) [para_into,5457.1.2,17.1.1] (x* (y*x))*rs(x,rs(z*y,z))=x.
5776 [para_into,5453.1.1.1,682.1.1] (x*y)*rs(z*rs(x,rs(u,y)),z)=x* (u*x).
8564 [para_into,5776.1.1.2.1,5559.1.1] (x*y)*rs(x,x* (z*x))=x* ((y*z)*x).
8576 (heat=1) [para_into,8564.1.1.2,13.1.1] (x*y)* (z*x)=x* ((y*z)*x).
```

For a delightful dessert, a comparison of the (final, for me) 15-step proof with the 23-step proof (from 2002) shows that ten of the equations in the 15-step proof are not present in the 23-step proof.

Next in order was a glance at proving Moufang 1 from Moufang 4, with the added constraints that neither Moufang 2 nor 3 be used in the proof. Indeed, a glance was all that was needed. I simply plucked the input file and proof from my research of 2002 and, with a short experiment, verified that I could do no better than the following.

A 3-Step Proof Deriving Moufang 1 From Moufang 4

```
----- Otter 3.2d, May 2002 -----
The process was started by wos on jaguar.mcs.anl.gov,
Thu Jun  6 12:15:45 2002
The command was "otter".  The process ID is 617.
----> UNIT CONFLICT at   0.08 sec ----> 1073 [binary,1072.1,10.1] $ANS(m1a).

Length of proof is 3.  Level of proof is 3.

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

7 [] x*1=x.
8 [] x* ((y*z)*x)= (x*y)* (z*x).
10 [] (a* (b*c))*a!= (a*b)* (c*a)|$ANS(m1a).
36 [] 1*x=x.
40 [para_into,8.1.1.2.1,7.1.1,flip.1] (x*y)* (1*x)=x* (y*x).
804 [para_into,40.1.2,8.1.1] (x* (y*z))* (1*x)= (x*y)* (z*x).
1072 (heat=1) [para_into,804.1.1.2,36.1.1] (x* (y*z))*x= (x*y)* (z*x).
```

The next natural move for this notebook would appear to be a discussion of proving from Moufang 1 Moufang 2. After all, I started with 3 implies 4, then 4 implies 1; therefore, 1 implies 2 would be the natural theorem to focus on next. But it must wait. Indeed, the story of 1 implies 2 is complicated and lengthy; that journey will be described as the last of the four concerning Moufang loops. Instead, I began with 2 implies 3, for which I chose a 26-step proof I had found in 2002, the following.

An Original 26-Step Proof Showing Moufang 2 Implies Moufang 3

```
----- Otter 3.2d, May 2002 -----
The process was started by wos on jaguar.mcs.anl.gov,
Thu Jun  6 09:51:23 2002
The command was "otter".  The process ID is 32569.
----> UNIT CONFLICT at  23.19 sec ----> 46815 [binary,46814.1,10.1] $ANS(m3a).

Length of proof is 26.  Level of proof is 17.

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

2 [] x*rs(x,y)=y.
4 [] ls(x,y)*y=x.
6 [] 1*x=x.
8 [] ((x*y)*z)*y=x* (y* (z*y)).
10 [] ((a*b)*a)*c!=a* (b* (a*c))|$ANS(m3a).
44 [] x*rs(x,y)=y.
45 [] rs(x,x*y)=y.
46 [] ls(x,y)*y=x.
47 [] ls(x*y,y)=x.
48 [] 1*x=x.
```

```
49 [] x*1=x.
53 [para_into,8.1.1.1.1,6.1.1,flip.1] 1* (x* (y*x))= (x*y)*x.
54 [para_into,8.1.1.1.1,4.1.1,flip.1] ls(x,y)* (y* (z*y))= (x*z)*y.
181 (heat=1) [para_into,53.1.1,48.1.1,flip.1] (x*y)*x=x* (y*x).
198 (heat=1) [para_into,54.1.1.2.2,46.1.1] ls(x,y)* (y*z)= (x*ls(z,y))*y.
215 (heat=1) [para_from,54.1.1,47.1.1.1] ls((x*y)*z,z* (y*z))=ls(x,z).
1938 [para_into,198.1.2.1,6.1.1] ls(1,x)* (x*y)=ls(y,x)*x.
2450 (heat=1) [para_into,1938.1.2,46.1.1] ls(1,x)* (x*y)=y.
4426 [para_into,2450.1.1.2,181.1.1] ls(1,x*y)* (x* (y*x))=x.
4435 [para_into,2450.1.1.2,2.1.1] ls(1,x)*y=rs(x,y).
4566 (heat=1) [para_from,4426.1.1,47.1.1.1] ls(x,x* (y*x))=ls(1,x*y).
4718 (heat=1) [para_into,4435.1.1,49.1.1,flip.1] rs(x,1)=ls(1,x).
8848 [para_into,4718.1.2,4566.1.2,flip.1] ls(x,x* (y*x))=rs(x*y,1).
8884 [para_from,4718.1.2,4435.1.1.1] rs(x,1)*y=rs(x,y).
8890 [para_from,4718.1.2,4.1.1.1] rs(x,1)*x=1.
11952 [para_from,8884.1.1,54.1.2.1] ls(rs(x,1),y)* (y* (z*y))=rs(x,z)*y.
12634 [para_from,8890.1.1,215.1.1.1.1] ls(1*x,x* (y*x))=ls(rs(y,1),x).
12979 (heat=1) [para_into,12634.1.1.1,48.1.1] ls(x,x* (y*x))=ls(rs(y,1),x).
17565 [para_into,12979.1.1,8848.1.1] rs(x*y,1)=ls(rs(y,1),x).
18410 [para_from,17565.1.2,11952.1.1.1] rs(x*y,1)* (x* (z*x))=rs(y,z)*x.
18769 [para_into,18410.1.1,8884.1.1] rs(x*y,x* (z*x))=rs(y,z)*x.
20223 (heat=1) [para_from,18769.1.1,44.1.1.2] (x*y)* (rs(y,z)*x)=x* (z*x).
28194 [para_into,20223.1.1.1,2.1.1] x* (rs(rs(y,x),z)*y)=y* (z*y).
28404 (heat=1) [para_into,28194.1.1.2.1,45.1.1,flip.1] x* ((rs(x,y)*z)*x)=y* (z*x).
28809 [para_from,20223.1.1,8.1.1.1,flip.1] x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y.
43091 [para_into,28809.1.1.2,28404.1.1,flip.1] (x* (y*x))*z=x* (y* (x*z)).
46814 [para_into,43091.1.1.1,181.1.2] ((x*y)*x)*z=x* (y* (x*z)).
```

An examination of this proof reveals that certain items appear more than once before deductions are shown. For example, clauses (2) and (44) are identical. You may naturally wonder about such occurrences, which I now discuss in part to fulfill my earlier promise concerning the use of the hot list strategy. The idea behind the introduction of the hot list strategy was the recognition that, when reading a proof from some text or paper on mathematics, certain items were continually visited. For an illustration, from an inspection of various proofs showing that rings in which the cube of x is x are commutative, one discovers that the key property $xxx = x$, is relied upon frequently. The thought, in the mid-1980s, was that perhaps an automated reasoning program would benefit by having (in effect) access to such a move. The hot list strategy was then introduced. In that strategy, items, formulas, or equations that are to be visited (so to speak) immediately are placed in a special list called the hot list. Indeed, with the hot list strategy, members of the hot list are automatically and immediately considered (by the inference rules in use) with each newly retained clause, *before* another conclusion is chosen to be the focus of attention to drive the program's reasoning. When you find, prior to the use of paramodulation, clauses (2) and (44) to be identical, you can correctly conclude that the corresponding equation has been placed in the (initial) hot list). (McCune, with much foresight, formulated the *dynamic hot list strategy*, a strategy that enables a program to adjoin, during the run, new items to the hot list.) When a clause is deduced and retained in the manner just described, it has heat-level 1. Further, reminiscent of recursion, with an assignment of the value 2 to the heat parameter, the heat-level-1 clauses (under discussion) that are retained will each immediately be considered by paramodulation with members of the hot list. As a useful aside, I note that by using the hot list strategy with a level-saturation approach, the program is able to look into levels that are greater than the level under consideration.

Now, to the high points of the journey, including an element of methodology I do not often use. The first significant development was based on a second 26-step proof, somewhat different from the 26-step proof just given. For the result in focus, I assigned the value 1 to the pick_given_ratio, the value 23 to max_weight, and the value 5 to max_distinct_vars. The first of the three choices was motivated by the intention of having OTTER strongly consider complex formulas retained early. The second choice was motivated by a guess, giving the program a fair amount of room in the context of complexity with regard to newly retained information. The third choice was motivated by the notion that equations would not be needed with six or more distinct variables present, but just perhaps five might cause an interesting equation to be placed in a proof. I also included in the weight_list(pick_given) twenty-six resonators corresponding to the new 26-step proof. To block the use of Moufang 1 and Moufang 4, I included a weight_list(purge_gen) with both forms of each equation, interchanging the arguments of each to get four elements of that weight list. I did use the value 1 for heat. The experiment in focus produced the following 24-step proof, a proof that turned out to play a key role in later experiments.

A 24-Step Proof Deriving Moufang 3 From Moufang 2

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Thu Jul 16 09:11:34 2009
The command was "otter".  The process ID is 7151.

----> UNIT CONFLICT at  35.27 sec ----> 109929 [binary,109928.1,10.1] $ANS(m3a).

Length of proof is 24.  Level of proof is 16.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
4 [] ls(x,y)*y=x.
6 [] 1*x=x.
7 [] x*1=x.
8 [] ((x*y)*z)*y=x* (y* (z*y)).
10 [] ((a*b)*a)*c!=a* (b* (a*c))|$ANS(m3a).
16 [] ((x*y)*z)*y=x* (y* (z*y)).
17 [] x*rs(x,y)=y.
18 [] rs(x,x*y)=y.
19 [] ls(x,y)*y=x.
21 [] 1*x=x.
22 [] x*1=x.
25 [para_into,8.1.1.1.1,6.1.1,flip.1] 1* (x* (y*x))= (x*y)*x.
26 [para_into,8.1.1.1.1,4.1.1,flip.1] ls(x,y)* (y* (z*y))= (x*z)*y.
31 [para_into,8.1.1.1,2.1.1,flip.1] x* (y* (rs(x*y,z)*y))=z*y.
92 (heat=1) [para_into,25.1.1,21.1.1,flip.1] (x*y)*x=x* (y*x).
106 (heat=1) [para_into,26.1.1.2.2,19.1.1] ls(x,y)* (y*z)= (x*ls(z,y))*y.
107 (heat=1) [para_into,26.1.1.2.2,17.1.1] ls(x,rs(y,z))* (rs(y,z)*z)= (x*y)*rs(y,z).
1998 [para_into,106.1.2.1,6.1.1] ls(1,x)* (x*y)=ls(y,x)*x.
2247 (heat=1) [para_into,1998.1.2,19.1.1] ls(1,x)* (x*y)=y.
2546 [para_into,107.1.1.2,7.1.1] ls(x,rs(y,1))*rs(y,1)= (x*y)*rs(y,1).
2586 (heat=1) [para_into,2546.1.1,19.1.1,flip.1] (x*y)*rs(y,1)=x.
7639 [para_into,2247.1.1.2,2.1.1] ls(1,x)*y=rs(x,y).
7959 (heat=1) [para_into,7639.1.1,22.1.1,flip.1] rs(x,1)=ls(1,x).
10124 [para_from,2586.1.1,3.1.1.2] rs(x*y,x)=rs(y,1).
16452 [para_from,7959.1.2,7639.1.1.1] rs(x,1)*y=rs(x,y).
18732 [para_into,16452.1.2,10124.1.1] rs(x*y,1)*x=rs(y,1).
23279 [para_from,18732.1.2,16452.1.1.1] (rs(x*y,1)*x)*z=rs(y,z).
23676 (heat=1) [para_from,23279.1.1,16.1.1.1,flip.1] rs(x*y,1)* (x* (z*x))=rs(y,z)*x.
86928 [para_into,23676.1.1,16452.1.1] rs(x*y,x* (z*x))=rs(y,z)*x.
```

88346 (heat=1) [para_from,86928.1.1,17.1.1.2] (x*y)* (rs(y,z)*x)=x* (z*x).
99715 [para_from,86928.1.1,31.1.1.2.2.1] x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y.
103756 [para_into,88346.1.1.1,2.1.1] x* (rs(rs(y,x),z)*y)=y* (z*y).
103842 (heat=1) [para_into,103756.1.1.2.1,18.1.1,flip.1] x* ((rs(x,y)*z)*x)=y* (z*x).
108108 [para_from,103842.1.1,99715.1.1.2,flip.1] (x* (y*x))*z=x* (y* (x*z)).
109928 [para_into,108108.1.1.1,92.1.2] ((x*y)*x)*z=x* (y* (x*z)).

I come now to the atypical methodological move referred to earlier. I was motivated by the disappointment at having found a proof only two steps shorter than that I had in hand in 2002. I was also motivated by the fact that two of my standard approaches to finding a shorter proof had yielded nothing of interest. The first of the approaches is based on blocking steps (of the proof in focus), one at a time or two at a time, with demodulation. The second of the two approaches relies on cramming. When cramming is the choice, one selects some subproof of some so-called intermediate step of the proof in focus and uses level saturation in the attempt to force or cram those steps into a shorter proof; the chosen proof steps are placed in the (initial) set of support, and level saturation is employed. Now, if a proof of length less than 24 was to be found on this trip—rather than simply trying a huge number of combinations of settings as assignments to parameters—the following idea occurred to mer

Specifically, I would turn again to cramming, but, rather than relying on a breadth-first search (level saturation), I would instead rely on McCune's ratio strategy, which blends complexity preference and first come first serve. I would select part of the 24-step proof, regardless of its being precisely a subproof of its last step, and place those steps in the initial set of support and see what resulted. In particular, I used the following input file.

An Input File for the Study of Moufang 2 with Nonstandard Cramming

```
% Sample Input File for the Study of Moufang Loops
% op(400, xfx,^).
% op(400, xfx, v).
op(400,xfx, *). % make all association explicit

% set(knuth_bendix).
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
set(ancestor_subsume).
set(back_sub).
% set(back_unit_deletion).
% assign(neg_weight,-5).
% set(print_proof_as_hints).

assign(max_weight,19).
% assign(change_limit_after,100).
% assign(new_max_weight,15).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).
% set(sos_queue).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).
```

```
assign(max_proofs,-1).
assign(pick_given_ratio,2).
assign(max_distinct_vars,3).
% assign(max_seconds,45).
assign(heat,0).

weight_list(pick_given).
% Following 26/15 prove the theorem
weight(1* (x* (y*x))= (x*y)*x,-6).
weight(ls(x,y)* (y* (z*y))= (x*z)*y,-6).
weight(x* (y* (rs(x*y,z)*y))=z*y,-6).
weight((x*y)*x=x* (y*x),-6).
weight(ls(x,y)* (y*z)= (x*ls(z,y))*y,-6).
weight(ls(x,rs(y,z))* (rs(y,z)*z)= (x*y)*rs(y,z),-6).
weight(ls(1,x)* (x*y)=ls(y,x)*x,-6).
weight(ls(1,x)* (x*y)=y,-6).
weight(ls(1,x)*y=rs(x,y),-6).
weight(rs(x,1)=ls(1,x),-6).
weight(rs(x,1)*y=rs(x,y),-6).
weight(ls(rs(x,1),y)* (y* (z*y))=rs(x,z)*y,-6).
weight(ls(rs(x,1),y)* (y* ((x*z)*y))=z*y,-6).
weight(ls(x,rs(y,1))*rs(y,1)= (x*y)*rs(y,1),-6).
weight((x*y)*rs(y,1)=x,-6).
weight(rs(x*y,x)=rs(y,1),-6).
weight(rs(x*y,1)*x=rs(y,1),-6).
weight(rs(x*y,1)=ls(rs(y,1),x),-6).
weight(ls(rs(x,1),y)*z=rs(y*x,z),-6).
weight(ls(rs(rs(x,y),1),x)*z=rs(y,z),-6).
weight(x* (y* ((ls(rs(y,1),x)*z)*y))=z*y,-6).
weight(x* (ls(rs(rs(y,x),1),y)*z)=z,-6).
weight(x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y,-6).
weight(x* ((rs(x,y)*z)*x)=y* (z*x),-6).
weight((x* (y*x))*z=x* (y* (x*z)),-6).
weight(((x*y)*x)*z=x* (y* (x*z)),-6).
% Following 20 prove step 14 of a 27-step, temp.moufang.2imp3.new.out1c.
weight(1* (x* (y*x))= (x*y)*x,-4).
weight(ls(x,y)* (y* (z*y))= (x*z)*y,-4).
weight((x*y)*x=x* (y*x),-4).
weight(ls(x,y)* (y*z)= (x*ls(z,y))*y,-4).
weight(ls((x*y)*z,z* (y*z))=ls(x,z),-4).
weight(ls(1,x)* (x*y)=ls(y,x)*x,-4).
weight(ls(1,x)* (x*y)=y,-4).
weight(ls(1,x*y)* (x* (y*x))=x,-4).
weight(ls(1,x)*y=rs(x,y),-4).
weight(ls(x,x* (y*x))=ls(1,x*y),-4).
weight(rs(x,1)=ls(1,x),-4).
weight(ls(x,x* (y*x))=rs(x*y,1),-4).
weight(rs(x,1)*y=rs(x,y),-4).
weight(rs(x,1)*x=1,-4).
weight(ls(1*x,x* (y*x))=ls(rs(y,1),x),-4).
weight(ls(x,x* (y*x))=ls(rs(y,1),x),-4).
weight(rs(x*y,1)=ls(rs(y,1),x),-4).
weight(ls(rs(x,1),y)*z=rs(y*x,z),-4).
```

```
weight(rs(x*y,x)=rs(y,1),-4).
weight((x*y)*rs(y,1)=x,-4).
% Following 9 from cramming on preceding 20 prove a very late step in a 25-step proof
weight(ls(rs(x,1),y)* (y* (z*y))=rs(x,z)*y,-4).
weight(ls(rs(x,1),y)* (y* ((x*z)*y))=z*y,-4).
weight(ls(rs(rs(x,y),1),x)*z=rs(y,z),-4).
weight(x* (ls(rs(rs(y,x),1),y)*z)=z,-4).
weight((x*y)* (ls(rs(y,1),x)*z)=z,-4).
weight(x* (y* ((ls(rs(y,1),x)*z)*y))=z*y,-4).
weight(x* ((rs(x,y)*z)*x)=y* (z*x),-4).
weight(x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y,-4).
weight((x* (y*x))*z=x* (y* (x*z)),-4).
end_of_list.

% The following list can be used to purge unwanted equations
weight_list(purge_gen).
% Blocking use of Moufang 1.
% weight(((x * y) * (z * x) = (x * (y * z)) * x),1000).
% Blocking use of Moufang 2.
% weight((((x * y) * z) * y = x * (y * (z * y)),1000).
% Blocking use of Moufang 3.
% weight((x * (y * (x * z)) = ((x * y) * x) * z),1000).
weight(junk,1000).
end_of_list.

% Used to complete applications of inference rules.
list(usable).
x = x.

x * rs(x,y) = y.  % right solvable
rs(x, x * y) = y.  % right solution is unique (implies left cancellation)
ls(x,y) * y = x.  % left solvable
ls(x * y , y) = x.  % left solution is unique (implies right cancellation)

% identity:
1 * x = x .
x * 1 = x.

% left cancellation
% x*y != u | x*z != u | y = z.
% right cancellation
% y*x != u | z*x != u | y = z.
end_of_list.

% Used to initiate applications of inference rules.
list(sos).
% Axiom, Moufang 2:
((x * y) * z) * y = x * (y * (z * y)).
% Following 12 of the 24 found here are being used for cramming
1* (x* (y*x))= (x*y)*x.
ls(x,y)* (y* (z*y))= (x*z)*y.
x* (y* (rs(x*y,z)*y))=z*y.
(x*y)*x=x* (y*x).
```

```
ls(x,y)* (y*z)= (x*ls(z,y))*y.
ls(x,rs(y,z))* (rs(y,z)*z)= (x*y)*rs(y,z).
ls(1,x)* (x*y)=ls(y,x)*x.
ls(1,x)* (x*y)=y.
ls(x,rs(y,1))*rs(y,1)= (x*y)*rs(y,1).
(x*y)*rs(y,1)=x.
ls(1,x)*y=rs(x,y).
rs(x,1)=ls(1,x).
% rs(x*y,x)=rs(y,1).
% rs(x,1)*y=rs(x,y).
% rs(x*y,1)*x=rs(y,1).
% (rs(x*y,1)*x)*z=rs(y,z).
% rs(x*y,1)* (x* (z*x))=rs(y,z)*x.
% rs(x*y,x* (z*x))=rs(y,z)*x.
% (x*y)* (rs(y,z)*x)=x* (z*x).
% x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y.
% x* (rs(rs(y,x),z)*y)=y* (z*y).
% x* ((rs(x,y)*z)*x)=y* (z*x).
% (x* (y*x))*z=x* (y* (x*z)).
% ((x*y)*x)*z=x* (y* (x*z)).
end_of_list.

% Used mainly to detect proof completion and to monitor progress.
list(passive).
% Following 24 negs for a 24-step proof that 2 imp 3 of Moufang.
1* (a* (b*a))!= (a*b)*a | $ANS(INTER).
ls(a,b)* (b* (c*b))!= (a*c)*b | $ANS(INTER).
a* (b* (rs(a*b,c)*b))!=c*b | $ANS(INTER).
(a*b)*a!=a* (b*a) | $ANS(INTER).
ls(a,b)* (b*c)!= (a*ls(c,b))*b | $ANS(INTER).
ls(a,rs(b,c))* (rs(b,c)*c)!= (a*b)*rs(b,c) | $ANS(INTER).
ls(1,a)* (a*b)!=ls(b,a)*a | $ANS(INTER).
ls(1,a)* (a*b)!=b | $ANS(INTER).
ls(a,rs(b,1))*rs(b,1)!= (a*b)*rs(b,1) | $ANS(INTER).
(a*b)*rs(b,1)!=a | $ANS(INTER).
ls(1,a)*b!=rs(a,b) | $ANS(INTER).
rs(a,1)!=ls(1,a) | $ANS(INTER).
rs(a*b,a)!=rs(b,1) | $ANS(INTER).
rs(a,1)*b!=rs(a,b) | $ANS(INTER).
rs(a*b,1)*a!=rs(b,1) | $ANS(INTER).
(rs(a*b,1)*a)*c!=rs(b,c) | $ANS(INTER).
rs(a*b,1)* (a* (c*a))!=rs(b,c)*a | $ANS(INTER).
rs(a*b,a* (c*a))!=rs(b,c)*a | $ANS(INTER).
(a*b)* (rs(b,c)*a)!=a* (c*a) | $ANS(INTER).
a* (b* ((rs(b,c)*a)*b))!= (a* (c*a))*b | $ANS(INTER).
a* (rs(rs(b,a),c)*b)!=b* (c*b) | $ANS(INTER).
a* ((rs(a,b)*c)*a)!=b* (c*a) | $ANS(INTER).
(a* (b*a))*c!=a* (b* (a*c)) | $ANS(INTER).
((a*b)*a)*c!=a* (b* (a*c)) | $ANS(INTER).
% Negation Axiom, Moufang 3:
a * (b * (a * c))  != ((a * b) * a) * c  |  $ANS(m3).
((a*b)*a)*c != a* (b* (a*c)) | $ANS(m3a).
end_of_list.
```

```
% The following list can be used to purge unwanted equations
list(demodulators).
%EQ(ls(x,x* (y*x))*z=rs(x*y,z),junk).
% Blocking use of Moufang 1.
EQ(((x * y) * (z * x) = (x * (y * z)) * x), junk).
EQ(((x * (y * z)) * x = ((x * y) * (z * x))), junk).
% Blocking use of Moufang 2.
% EQ((((x * y) * z) * y = x * (y * (z * y))), junk).
% EQ(((x * (y * (z * y))) = (((x * y) * z) * y)), junk).
% Blocking use of Moufang 3.
% EQ((x * (y * (x * z)) = ((x * y) * x) * z), junk).
% EQ((((x * y) * x) * z = x * (y * (x * z))), junk).
% Blocking use of Moufang 4.
EQ((x * ((y * z) * x) = (x * y) * (z * x)), junk).
EQ(((x * y) * (z * x) = x * ((y * z) * x)), junk).
end_of_list.

list(hints2).
% Following 149 borrowed from temp.moufang.3imp4.new.in1a
x*rs(x,y)=y.
y=x*rs(x,y).
rs(x,x*y)=y.
y=rs(x,x*y).
ls(x,y)*y=x.
x=ls(x,y)*y.
ls(x*y,y)=x.
x=ls(x*y,y).
1*x=x.
x=1*x.
x*1=x.
x=x*1.
x=x.
x=x.
x* (y* (x*z))= ((x*y)*x)*z.
((x*y)*x)*z=x* (y* (x*z)).
a* ((b*c)*a)!= (a*b)* (c*a).
(a*b)* (c*a)!=a* ((b*c)*a).
((x*rs(x,y))*x)*z=x* (rs(x,y)* (x*z)).
x* (rs(x,y)* (x*z))= ((x*rs(x,y))*x)*z.
(x*y)*z=y* (rs(y,x)* (y*z)).
y* (rs(y,x)* (y*z))= (x*y)*z.
x* (rs(x,y)* (x*z))= (y*x)*z.
(y*x)*z=x* (rs(x,y)* (x*z)).
((x*y)*x)*1= (x*y)*x.
(x*y)*x= ((x*y)*x)*1.
((x*y)*x)*1=x* (y* (x*1)).
x* (y* (x*1))= ((x*y)*x)*1.
(x*y)*x=x* (y* (x*1)).
x* (y* (x*1))= (x*y)*x.
(x*y)*x=x* (y*x).
x* (y*x)= (x*y)*x.
x* (rs(x,y)* (x*rs(x,z)))= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)* (x*rs(x,z))).
```

x* (rs(x,y)*z)= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)*z).
(x*y)*rs(y,z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)*rs(y,z).
(1*x)*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)= (1*x)*rs(x,y).
x*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)=x*rs(x,y).
x=y* (rs(y,1)*x).
y* (rs(y,1)*x)=x.
x* (rs(x,1)*y)=y.
y=x* (rs(x,1)*y).
(ls(x,y)*y)*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)= (ls(x,y)*y)*rs(y,z).
x*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)=x*rs(y,z).
(x*rs(x,y))*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))*rs(rs(x,y),z).
x*rs(rs(y,x),z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x*rs(rs(y,x),z).
ls((x*y)*rs(y,z),rs(y,z))=x*y.
x*y=ls((x*y)*rs(y,z),rs(y,z)).
ls(x* (rs(x,y)*z),rs(x,z))=y*x.
y*x=ls(x* (rs(x,y)*z),rs(x,z)).
x* (rs(x,ls(y,x))*z)=y*rs(x,z).
y*rs(x,z)=x* (rs(x,ls(y,x))*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z).
y*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,ls(y,x))*1=rs(x,ls(y,x)).
rs(x,ls(y,x))=rs(x,ls(y,x))*1.
x* (rs(x,ls(y,x))*1)=y*rs(x,1).
y*rs(x,1)=x* (rs(x,ls(y,x))*1).
x*rs(x,ls(y,x))=y*rs(x,1).
y*rs(x,1)=x*rs(x,ls(y,x)).
x*rs(x,ls(y,x))=ls(y,x).
ls(y,x)=x*rs(x,ls(y,x)).
ls(x,y)=x*rs(y,1).
x*rs(y,1)=ls(x,y).
rs(rs(x,y),x)*1=rs(rs(x,y),x).
rs(rs(x,y),x)=rs(rs(x,y),x)*1.
rs(x,y)* (rs(rs(x,y),x)*1)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)* (rs(rs(x,y),x)*1).
rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)*rs(rs(x,y),x).
rs(x,y)*rs(rs(x,y),x)=x.
x=rs(x,y)*rs(rs(x,y),x).
x=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=x.
x*rs(rs(y,x),1)=y.
y=x*rs(rs(y,x),1).
ls(x* (rs(x,y)*z),rs(x,z))= (x* (rs(x,y)*z))*rs(rs(x,z),1).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=ls(x* (rs(x,y)*z),rs(x,z)).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=y*x.

y*x= (x* (rs(x,y)*z))*rs(rs(x,z),1).
rs(x,y)*rs(rs(x,y),z)=z.
z=rs(x,y)*rs(rs(x,y),z).
(x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1)=y*x.
y*x= (x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
z*x= (x*y)*rs(rs(x,rs(rs(x,z),y)),1).
rs(x,x*rs(rs(y,x),1))=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,x*rs(rs(y,x),1)).
rs(x,y)=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,y).
rs(rs(x,y),1)=rs(y,x).
rs(y,x)=rs(rs(x,y),1).
rs(rs(x,rs(rs(x,y),z)),1)=rs(rs(rs(x,y),z),x).
rs(rs(rs(x,y),z),x)=rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(rs(x,z),y),x)=z*x.
z*x= (x*y)*rs(rs(rs(x,z),y),x).
(x*y)*rs(rs(rs(x,x*z),y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(rs(x,x*z),y),x).
(x*y)*rs(rs(z,y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(z,y),x).
(x*y)*rs(rs(z,y),x)=x* (z*x).
x* (z*x)= (x*y)*rs(rs(z,y),x).
x* (rs(x,y)* (x* (rs(x,1)*z)))= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)* (x* (rs(x,1)*z))).
x* (rs(x,y)*z)= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)*z).
(x*y)* (rs(y,1)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)* (rs(y,1)*z).
(x*rs(x,y))* (rs(rs(x,y),1)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))* (rs(rs(x,y),1)*z).
x* (rs(rs(y,x),1*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(rs(y,x),1)*z).
x* (rs(x,y)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(x,y)*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
x* (rs(x,y)*z)=x*rs(rs(y,x),z).
x*rs(rs(y,x),z)=x* (rs(x,y)*z).
rs(x,x*rs(rs(y,x),z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x*rs(rs(y,x),z)).
rs(x,x* (rs(x,y)*z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x* (rs(x,y)*z)).
rs(x,x* (rs(x,y)*z))=rs(x,y)*z.
rs(x,y)*z=rs(x,x* (rs(x,y)*z)).
rs(x,y)*z=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,y)*z.
rs(rs(x,y),z)=rs(y,x)*z.
rs(y,x)*z=rs(rs(x,y),z).
(x*y)* (rs(y,z)*x)=x* (z*x).
x* (z*x)= (x*y)* (rs(y,z)*x).
(x*y)* (rs(y,y*z)*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (rs(y,y*z)*x).

```
(x*y)* (z*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (z*x).
a* ((b*c)*a)= (a*b)* (c*a).
(a*b)* (c*a)=a* ((b*c)*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
$F.
% Following 27/17 prove 2 imp 3 for Moufang without demod
1* (x* (y*x))= (x*y)*x.
ls(x,y)* (y* (z*y))= (x*z)*y.
(x*y)*x=x* (y*x).
ls(x,y)* (y*z)= (x*ls(z,y))*y.
ls((x*y)*z,z* (y*z))=ls(x,z).
ls(1,x)* (x*y)=ls(y,x)*x.
ls(1,x)* (x*y)=y.
ls(1,x*y)* (x* (y*x))=x.
ls(1,x)*y=rs(x,y).
ls(x,x* (y*x))=ls(1,x*y).
rs(x,1)=ls(1,x).
ls(x,x* (y*x))*z=rs(x*y,z).
ls(x,x* (y*x))=rs(x*y,1).
rs(x,1)*y=rs(x,y).
rs(x,1)*x=1.
ls(rs(x,1),y)* (y* (z*y))=rs(x,z)*y.
ls(1*x,x* (y*x))=ls(rs(y,1),x).
ls(x,x* (y*x))=ls(rs(y,1),x).
rs(x*y,1)=ls(rs(y,1),x).
rs(x*y,1)* (x* (z*x))=rs(y,z)*x.
rs(x*y,x* (z*x))=rs(y,z)*x.
(x*y)* (rs(y,z)*x)=x* (z*x).
x* (rs(rs(y,x),z)*y)=y* (z*y).
x* ((rs(x,y)*z)*x)=y* (z*x).
x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y.
(x* (y*x))*z=x* (y* (x*z)).
((x*y)*x)*z=x* (y* (x*z)).
end_of_list.

% Used for the hot list strategy.
list(hot).
% Axiom, Moufang 1:
% (x * y) * (z * x) = (x * (y * z)) * x.
% Axiom, Moufang 2:
((x * y) * z) * y = x * (y * (z * y)).
% Axiom, Moufang 3:
% x * (y * (x * z)) = ((x * y) * x) * z.
% Axiom, Moufang 4:
% x * ((y * z) * x) = (x * y) * (z * x).
x * rs(x,y) = y. % right solvable
rs(x, x * y) = y. % right solution is unique (implies left cancellation)
ls(x,y) * y = x. % left solvable
ls(x * y , y) = x. % left solution is unique (implies right cancellation)
```

```
% identity:
1 * x = x.
x * 1 = x.
end_of_list.
```

With this input file, in addition to the twelve steps I had adjoined to list(sos), OTTER found a 14-step proof of Moufang 3. Seven of the fourteen steps are not among those of the 24-step proof that prompted this experiment. Therefore, put another way, five of the twenty-four deduced steps of the best proof I had so far were not among the twenty-six, the twelve plus the fourteen.

On the surface, you might fear, as I did, that the use of these (new) twenty-six steps as resonators (together with twenty-six the earlier run that yielded the 26-step proof in 2002) would merely return the (in effect) newest 26-step proof, and the so-called sojourn would be terminated. I did assign a much smaller value to the first twenty-six resonators than I had to the second. As it turned out, OTTER found two proofs, the first (a so-called newest) of length 26 and the second, quite a bit later, of length 23. (A later experiment strongly suggests that the twenty-six resonators corresponding to the 2002 proof were crucial to finding the 23-step proof; I also am fairly certain that those old hints from 2002 played an important role.) You thus have a nice example of how research from seven years ago aided research today, in mid-2009. This part of the journey may provide yet one more approach to either finding a better proof than that in hand or finding a first proof; the approach focuses on using resonators or hints that yielded success in earlier studies.

As you would correctly predict, I then made one more experiment immediately: I took the file that had yielded the 23-step proof and replaced all of its resonators by twenty-three corresponding to the 23-step proof just found. I shortly give the input file and the resulting 20-step proof, noting that OTTER first found (as expected, but not always the case) a 23-step proof,

An Input File Yielding a 20-Step Proof of Moufang 3 from Moufang 2

```
% Sample Input File for the Study of Moufang Loops
% op(400, xfx, ^).
% op(400, xfx, v).
op(400,xfx, *). % make all association explicit

% set(knuth_bendix).
set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
set(ancestor_subsume).
set(back_sub).
% set(back_unit_deletion).
% assign(neg_weight,-5).
% set(print_proof_as_hints).

assign(max_weight,19).
% assign(change_limit_after,100).
% assign(new_max_weight,15).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).
% set(sos_queue).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).
```

```
assign(max_proofs,-1).
assign(pick_given_ratio,2).
assign(max_distinct_vars,3).
% assign(max_seconds,45).
assign(heat,0).

weight_list(pick_given).
% Following 23/13 prove Moufang 3 from 2, temp.moufang.2imp3.new.ot1j
weight(1* (x* (y*x))= (x*y)*x,-10).
weight(ls(x,y)* (y* (z*y))= (x*z)*y,-10).
weight((x*y)*x=x* (y*x),-10).
weight(ls(x,y)* (y*z)= (x*ls(z,y))*y,-10).
weight(ls(x,rs(y,z))* (rs(y,z)*z)= (x*y)*rs(y,z),-10).
weight(ls(1,x)* (x*y)=ls(y,x)*x,-10).
weight(ls(x,rs(y,1))*rs(y,1)= (x*y)*rs(y,1),-10).
weight(ls(1,x)* (x*y)=y,-10).
weight((x*y)*rs(y,1)=x,-10).
weight(ls(1,x)*y=rs(x,y),-10).
weight(rs(x*y,x)=rs(y,1),-10).
weight(rs(x,1)=ls(1,x),-10).
weight(ls(1,x)* (y* (z*y))= (rs(x,y)*z)*y,-10).
weight(x* (ls(1,x)*y)=y,-10).
weight(rs(x,1)*y=rs(x,y),-10).
weight(rs(x*y,x)*z=rs(y,z),-10).
weight(x* ((rs(x,y)*z)*y)=y* (z*y),-10).
weight((x*y)* (rs(y,z)*x)=x* (z*x),-10).
weight(x* (rs(rs(y,x),z)*y)=y* (z*y),-10).
weight(x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y,-10).
weight(x* ((rs(x,y)*z)*x)=y* (z*x),-10).
weight((x* (y*x))*z=x* (y* (x*z)),-10).
weight(((x*y)*x)*z=x* (y* (x*z)),-10).
end_of_list.

% The following list can be used to purge unwanted equations of various types.
weight_list(purge_gen).
% Blocking use of Moufang 1.
% weight(((x * y) * (z * x) = (x * (y * z)) * x),1000).
% Blocking use of Moufang 2.
% weight((((x * y) * z) * y = x * (y * (z * y)),1000).
% Blocking use of Moufang 3.
% weight((x * (y * (x * z)) = ((x * y) * x) * z),1000).
weight(junk,1000).
end_of_list.

% Used to complete applications of inference rules.
list(usable).
x = x.

x * rs(x,y) = y. % right solvable
rs(x, x * y) = y. % right solution is unique (implies left cancellation)
ls(x,y) * y = x. % left solvable
ls(x * y , y) = x. % left solution is unique (implies right cancellation)
```

```
% identity:
1 * x = x .
x * 1 = x.

% left cancellation
% x*y != u | x*z != u | y = z.
% right cancellation
% y*x != u | z*x != u | y = z.
end_of_list.

% Used to initiate applications of inference rules.
list(sos).
% Axiom, Moufang 2:
((x * y) * z) * y = x * (y * (z * y)).
end_of_list.

% Used mainly to detect proof completion and to monitor progress.
list(passive).
% Following 24 negs for a 24-step proof that 2 imp 3 of Moufang.
% 1* (a* (b*a))!= (a*b)*a | $ANS(INTER).
% ls(a,b)* (b* (c*b))!= (a*c)*b | $ANS(INTER).
% a* (b* (rs(a*b,c)*b))!=c*b | $ANS(INTER).
% (a*b)*a!=a* (b*a) | $ANS(INTER).
% ls(a,b)* (b*c)!= (a*ls(c,b))*b | $ANS(INTER).
% ls(a,rs(b,c))* (rs(b,c)*c)!= (a*b)*rs(b,c) | $ANS(INTER).
% ls(1,a)* (a*b)!=ls(b,a)*a | $ANS(INTER).
% ls(1,a)* (a*b)!=b | $ANS(INTER).
% ls(a,rs(b,1))*rs(b,1)!= (a*b)*rs(b,1) | $ANS(INTER).
% (a*b)*rs(b,1)!=a | $ANS(INTER).
% ls(1,a)*b!=rs(a,b) | $ANS(INTER).
% rs(a,1)!=ls(1,a) | $ANS(INTER).
% rs(a*b,a)!=rs(b,1) | $ANS(INTER).
% rs(a,1)*b!=rs(a,b) | $ANS(INTER).
% rs(a*b,1)*a!=rs(b,1) | $ANS(INTER).
% (rs(a*b,1)*a)*c!=rs(b,c) | $ANS(INTER).
% rs(a*b,1)* (a* (c*a))!=rs(b,c)*a | $ANS(INTER).
% rs(a*b,a* (c*a))!=rs(b,c)*a | $ANS(INTER).
(a*b)* (rs(b,c)*a)!=a* (c*a) | $ANS(INTER).
a* (b* ((rs(b,c)*a)*b))!= (a* (c*a))*b | $ANS(INTER).
a* (rs(rs(b,a),c)*b)!=b* (c*b) | $ANS(INTER).
a* ((rs(a,b)*c)*a)!=b* (c*a) | $ANS(INTER).
(a* (b*a))*c!=a* (b* (a*c)) | $ANS(INTER).
((a*b)*a)*c!=a* (b* (a*c)) | $ANS(INTER).
% Negation Axiom, Moufang 3:
a * (b * (a * c)) != ((a * b) * a) * c | $ANS(m3).
((a*b)*a)*c != a* (b* (a*c)) | $ANS(m3a).
% ((a * b) * (c * a) != (a * (b * c)) * a) | $ANS(m1).
% ((a * b) * c) * b != a * (b * (c * b)) | $ANS(m2).
% a * (b * (a * c)) != ((a * b) * a) * c | $ANS(m3).
% a * ((b * c) * a) != (a * b) * (c * a) | $ANS(m4).
end_of_list.

% The following list can be used to purge unwanted equations of various types.
```

```
list(demodulators).
%EQ(ls(x,x* (y*x))*z=rs(x*y,z),junk).
%  Blocking use of Moufang 1.
EQ(((x * y) * (z * x) = (x * (y * z)) * x), junk).
EQ(((x * (y * z)) * x = ((x * y) * (z * x))), junk).
%  Blocking use of Moufang 2.
%  EQ((((x * y) * z) * y = x * (y * (z * y))), junk).
%  EQ(((x * (y * (z * y))) = (((x * y) * z) * y)), junk).
%  Blocking use of Moufang 3.
%  EQ((x * (y * (x * z)) = ((x * y) * x) * z), junk).
%  EQ((((x * y) * x) * z = x * (y * (x * z))), junk).
%  Blocking use of Moufang 4.
EQ((x * ((y * z) * x) = (x * y) * (z * x)), junk).
EQ(((x * y) * (z * x) = x * ((y * z) * x)), junk).
end_of_list.

list(hints2).
%  Following 149 borrowed from temp.moufang.3imp4.new.in1a
x*rs(x,y)=y.
y=x*rs(x,y).
rs(x,x*y)=y.
y=rs(x,x*y).
ls(x,y)*y=x.
x=ls(x,y)*y.
ls(x*y,y)=x.
x=ls(x*y,y).
1*x=x.
x=1*x.
x*1=x.
x=x*1.
x=x.
x=x.
x* (y* (x*z))= ((x*y)*x)*z.
((x*y)*x)*z=x* (y* (x*z)).
a* ((b*c)*a)!= (a*b)* (c*a).
(a*b)* (c*a)!=a* ((b*c)*a).
((x*rs(x,y))*x)*z=x* (rs(x,y)* (x*z)).
x* (rs(x,y)* (x*z))= ((x*rs(x,y))*x)*z.
(x*y)*z=y* (rs(y,x)* (y*z)).
y* (rs(y,x)* (y*z))= (x*y)*z.
x* (rs(x,y)* (x*z))= (y*x)*z.
(y*x)*z=x* (rs(x,y)* (x*z)).
((x*y)*x)*1= (x*y)*x.
(x*y)*x= ((x*y)*x)*1.
((x*y)*x)*1=x* (y* (x*1)).
x* (y* (x*1))= ((x*y)*x)*1.
(x*y)*x=x* (y* (x*1)).
x* (y* (x*1))= (x*y)*x.
(x*y)*x=x* (y*x).
x* (y*x)= (x*y)*x.
x* (rs(x,y)* (x*rs(x,z)))= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)* (x*rs(x,z))).
x* (rs(x,y)*z)= (y*x)*rs(x,z).
```

(y*x)*rs(x,z)=x* (rs(x,y)*z).
(x*y)*rs(y,z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)*rs(y,z).
(1*x)*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)= (1*x)*rs(x,y).
x*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)=x*rs(x,y).
x=y* (rs(y,1)*x).
y* (rs(y,1)*x)=x.
x* (rs(x,1)*y)=y.
y=x* (rs(x,1)*y).
(ls(x,y)*y)*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)= (ls(x,y)*y)*rs(y,z).
x*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)=x*rs(y,z).
(x*rs(x,y))*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))*rs(rs(x,y),z).
x*rs(rs(y,x),z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x*rs(rs(y,x),z).
ls((x*y)*rs(y,z),rs(y,z))=x*y.
x*y=ls((x*y)*rs(y,z),rs(y,z)).
ls(x* (rs(x,y)*z),rs(x,z))=y*x.
y*x=ls(x* (rs(x,y)*z),rs(x,z)).
x* (rs(x,ls(y,x))*z)=y*rs(x,z).
y*rs(x,z)=x* (rs(x,ls(y,x))*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z).
y*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,ls(y,x))*1=rs(x,ls(y,x)).
rs(x,ls(y,x))=rs(x,ls(y,x))*1.
x* (rs(x,ls(y,x))*1)=y*rs(x,1).
y*rs(x,1)=x* (rs(x,ls(y,x))*1).
x*rs(x,ls(y,x))=y*rs(x,1).
y*rs(x,1)=x*rs(x,ls(y,x)).
x*rs(x,ls(y,x))=ls(y,x).
ls(y,x)=x*rs(x,ls(y,x)).
ls(x,y)=x*rs(y,1).
x*rs(y,1)=ls(x,y).
rs(rs(x,y),x)*1=rs(rs(x,y),x).
rs(rs(x,y),x)=rs(rs(x,y),x)*1.
rs(x,y)* (rs(rs(x,y),x)*1)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)* (rs(rs(x,y),x)*1).
rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)*rs(rs(x,y),x).
rs(x,y)*rs(rs(x,y),x)=x.
x=rs(x,y)*rs(rs(x,y),x).
x=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=x.
x*rs(rs(y,x),1)=y.
y=x*rs(rs(y,x),1).
ls(x* (rs(x,y)*z),rs(x,z))= (x* (rs(x,y)*z))*rs(rs(x,z),1).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=ls(x* (rs(x,y)*z),rs(x,z)).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=y*x.
y*x= (x* (rs(x,y)*z))*rs(rs(x,z),1).

rs(x,y)*rs(rs(x,y),z)=z.
z=rs(x,y)*rs(rs(x,y),z).
(x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1)=y*x.
y*x= (x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
z*x= (x*y)*rs(rs(x,rs(rs(x,z),y)),1).
rs(x,x*rs(rs(y,x),1))=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,x*rs(rs(y,x),1)).
rs(x,y)=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,y).
rs(rs(x,y),1)=rs(y,x).
rs(y,x)=rs(rs(x,y),1).
rs(rs(x,rs(rs(x,y),z)),1)=rs(rs(rs(x,y),z),x).
rs(rs(rs(x,y),z),x)=rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(rs(x,z),y),x)=z*x.
z*x= (x*y)*rs(rs(rs(x,z),y),x).
(x*y)*rs(rs(rs(x,x*z),y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(rs(x,x*z),y),x).
(x*y)*rs(rs(z,y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(z,y),x).
(x*y)*rs(rs(z,y),x)=x* (z*x).
x* (z*x)= (x*y)*rs(rs(z,y),x).
x* (rs(x,y)* (x* (rs(x,1)*z)))= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)* (x* (rs(x,1)*z))).
x* (rs(x,y)*z)= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)*z).
(x*y)* (rs(y,1)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)* (rs(y,1)*z).
(x*rs(x,y))* (rs(rs(x,y),1)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))* (rs(rs(x,y),1)*z).
x* (rs(rs(y,x),1)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(rs(y,x),1)*z).
x* (rs(x,y)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(x,y)*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
x* (rs(x,y)*z)=x*rs(rs(y,x),z).
x*rs(rs(y,x),z)=x* (rs(x,y)*z).
rs(x,x*rs(rs(y,x),z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x*rs(rs(y,x),z)).
rs(x,x* (rs(x,y)*z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x* (rs(x,y)*z)).
rs(x,x* (rs(x,y)*z))=rs(x,y)*z.
rs(x,y)*z=rs(x,x* (rs(x,y)*z)).
rs(x,y)*z=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,y)*z.
rs(rs(x,y),z)=rs(y,x)*z.
rs(y,x)*z=rs(rs(x,y),z).
(x*y)* (rs(y,z)*x)=x* (z*x).
x* (z*x)= (x*y)* (rs(y,z)*x).
(x*y)* (rs(y,y*z)*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (rs(y,y*z)*x).
(x*y)* (z*x)=x* ((y*z)*x).

```
x* ((y*z)*x)= (x*y)* (z*x).
a* ((b*c)*a)= (a*b)* (c*a).
(a*b)* (c*a)=a* ((b*c)*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
$F.
% Following 27/17 prove 2 imp 3 for Moufang without demod and forward, from temp.loop23.out5
1* (x* (y*x))= (x*y)*x.
ls(x,y)* (y* (z*y))= (x*z)*y.
(x*y)*x=x* (y*x).
ls(x,y)* (y*z)= (x*ls(z,y))*y.
ls((x*y)*z,z* (y*z))=ls(x,z).
ls(1,x)* (x*y)=ls(y,x)*x.
ls(1,x)* (x*y)=y.
ls(1,x*y)* (x* (y*x))=x.
ls(1,x)*y=rs(x,y).
ls(x,x* (y*x))=ls(1,x*y).
rs(x,1)=ls(1,x).
ls(x,x* (y*x))*z=rs(x*y,z).
ls(x,x* (y*x))=rs(x*y,1).
rs(x,1)*y=rs(x,y).
rs(x,1)*x=1.
ls(rs(x,1),y)* (y* (z*y))=rs(x,z)*y.
ls(1*x,x* (y*x))=ls(rs(y,1),x).
ls(x,x* (y*x))=ls(rs(y,1),x).
rs(x*y,1)=ls(rs(y,1),x).
rs(x*y,1)* (x* (z*x))=rs(y,z)*x.
rs(x*y,x* (z*x))=rs(y,z)*x.
(x*y)* (rs(y,z)*x)=x* (z*x).
x* (rs(rs(y,x),z)*y)=y* (z*y).
x* ((rs(x,y)*z)*x)=y* (z*x).
x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y.
(x* (y*x))*z=x* (y* (x*z)).
((x*y)*x)*z=x* (y* (x*z)).
end_of_list.

% Used for the hot list strategy.
list(hot).
% Axiom, Moufang 1:
% (x * y) * (z * x) = (x * (y * z)) * x.
% Axiom, Moufang 2:
((x * y) * z) * y = x * (y * (z * y)).
% Axiom, Moufang 3:
% x * (y * (x * z)) = ((x * y) * x) * z.
% Axiom, Moufang 4:
% x * ((y * z) * x) = (x * y) * (z * x).
x * rs(x,y) = y. % right solvable
rs(x, x * y) = y. % right solution is unique (implies left cancellation)
ls(x,y) * y = x. % left solvable
ls(x * y , y) = x. % left solution is unique (implies right cancellation)
% identity:
```

```
1 * x = x.
x * 1 = x.
end_of_list.
```

A 20-Step Proof of Moufang 3 from Moufang 2

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Thu Jul 30 17:34:56 2009
The command was "otter".  The process ID is 3173.

----> UNIT CONFLICT at 1289.45 sec ----> 756887 [binary,756886.1,16.1] $ANS(m3a).

Length of proof is 20.  Level of proof is 13.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
4 [] ls(x,y)*y=x.
6 [] 1*x=x.
8 [] ((x*y)*z)*y=x* (y* (z*y)).
16 [] ((a*b)*a)*c!=a* (b* (a*c))|$ANS(m3a).
205 [para_into,8.1.1.1.1,6.1.1,flip.1] 1* (x* (y*x))= (x*y)*x.
206 [para_into,8.1.1.1.1,4.1.1,flip.1] ls(x,y)* (y* (z*y))= (x*z)*y.
230 [para_into,205.1.1,6.1.1,flip.1] (x*y)*x=x* (y*x).
244 [para_into,206.1.1.2.2,4.1.1] ls(x,y)* (y*z)= (x*ls(z,y))*y.
332 [para_into,244.1.2.1,6.1.1] ls(1,x)* (x*y)=ls(y,x)*x.
447 [para_into,332.1.2,4.1.1] ls(1,x)* (x*y)=y.
534 [para_into,447.1.1.2,2.1.1] ls(1,x)*y=rs(x,y).
547 [para_from,447.1.1,206.1.1.2] ls(x,ls(1,y))*ls(1,y)= (x*y)*ls(1,y).
786 [para_from,534.1.1,8.1.1.1.1,flip.1] ls(1,x)* (y* (z*y))= (rs(x,y)*z)*y.
794 [para_from,534.1.2,2.1.1.2] x* (ls(1,x)*y)=y.
1159 [para_into,794.1.1.2,786.1.1] x* ((rs(x,y)*z)*y)=y* (z*y).
114880 [para_into,547.1.1,4.1.1,flip.1] (x*y)*ls(1,y)=x.
376342 [para_from,114880.1.1,3.1.1.2] rs(x*y,x)=ls(1,y).
749217 [para_from,376342.1.2,534.1.1.1] rs(x*y,x)*z=rs(y,z).
751021 [para_from,749217.1.1,1159.1.1.2.1] (x*y)* (rs(y,z)*x)=x* (z*x).
751591 [para_into,751021.1.1.1,2.1.1] x* (rs(rs(y,x),z)*y)=y* (z*y).
752234 [para_from,751021.1.1,8.1.1.1,flip.1] x* (y* ((rs(y,z)*x)*y))= (x* (z*x))*y.
753381 [para_into,751591.1.1.2.1,3.1.1,flip.1] x* ((rs(x,y)*z)*x)=y* (z*x).
756208 [para_from,753381.1.1,752234.1.1.2,flip.1] (x* (y*x))*z=x* (y* (x*z)).
756886 [para_into,756208.1.1.1,230.1.2] ((x*y)*x)*z=x* (y* (x*z)).
```

Of interest, certainly to me, is the fact that the 20-step proof contains seven equations not present in the original 26-step proof from 2002. I cannot easily say why these seven years later I was able to make such progress.

By way of a postscript, the same 20-step proof is found if, in addition to the cited twenty-six resonators, you include twenty-six corresponding to the 26-step proof obtained in 2002. Again, if you do so and assign a value to each that is larger than the value assigned to the twenty-six I used, all goes well. I typically discard resonators that I used in one experiment when running a later one based on having found an even shorter proof. I do so because I have noticed that the presence of older resonators can sometimes cause the program to simply return an older proof.

And now I come to the complicated story focusing on deducing Moufang 2 from Moufang 1, pure with respect to both Moufang 3 and Moufang 4. This last story is vaguely reminiscent of what sometimes occurs in mathematics: A mathematician asserts that, if certain lemmas can be proved, then the theorem of concern will yield its treasure and also be proved. In particular, when I began my study of proving Moufang 2 from Moufang 1, with the constraint that neither Moufang 3 nor 4 be allowed to participate, I had in hand two purported proofs. The first proof, one of length 18, was unsatisfactory in that Moufang 2 (the target) had participated, entering the picture because of being inadvertently placed in a hot list; I earlier issued a warning about such a possibility. The second proof, one of length 29, was unsatisfactory in that it relied in various places on demodulation and was (to some extent) bidirectional; both features merit comment. Demodulation was formulated to canonicalize and simplify, thus removing a type of redundancy from the space of retained information. A Knuth-Bendix approach relies on demodulation. Unfortunately, at least with OTTER, when a proof contains occurrences of demodulation, they are not contained when the proof length is presented. Therefore, a cursory glance at proof length can be misleading, especially when demodulation occurs frequently in the deduced steps. As for a bidirectional proof, I have in mind a proof that proceeds forward from the axioms, usually using them to complete an application of a inference rule with the so-called special hypothesis driving the search, and at the same time backward from the negation or denial of the theorem to be proved. I always prefer, in the best case, that all deduced steps be drawn without reliance on the negation of the target, having that negation come into play at, say, unit conflict. When a conjunction is the target, I also prefer that none of the negated members play a direct role in the deduced steps. Therefore, the proof length measures, in the best case for me, only steps that are traceable to the problem description, not counting the denial of the theorem. To connect the so-called flaws of the two proofs I had before me with the earlier remark concerning the assertion of being able to complete the desired proof if various lemmas could be proved, you could view the two flawed proofs I had as containing useful lemmas with additional lemmas needed. So, with this preamble in hand, I now focus on the intricate paths of the labyrinth whose goal was a proof of Moufang 2 from 1, pure with respect to 3 and 4.

If you prefer to avoid sharing the obvious frustration I experienced before I finally mined the large gold nugget I sought, simply turn the pages until you find a proof that deduces Moufang 2 from Moufang 1. In my first few experiments, I permitted OTTER to apply ancestor subsumption in case some short proof was found, but soon discontinued its use for quite a while because that procedure eats CPU time. I began in what might be termed a naive or an optimistic frame of mind. Specifically, I placed the negations of the steps of the 18-step proof in list(passive), with the notion that perhaps OTTER could somehow find a proof. Yes, I had noticed that the sixteenth step of the 18-step proof had as a parent the unwanted Moufang 2 from the hot list, which (as noted) was clearly not allowed. That sixteenth step was used as a parent for the eighteenth step. I chose to try to fill in the gaps of the 18-step proof, perhaps obtaining a longer proof, because the other proof I had in hand appeared to present bigger obstacles: relying on demodulation and being a bidirectional proof. Of course, the passive list also contained two versions of the target, Moufang 2, the second obtained by interchanging its arguments; both forms were in the negated form to permit the detection, if and when such occurred, of unit conflict with the goal. A promising start: OTTER quickly found proofs of the first fifteen steps of the 18-step proof, and it also found a proof of the seventeenth step. Although I was not aware of what was happening at the time, I now entered a maze. I tried to complete the desired proof; I turned to cramming, adjoining to the list(sos) fifteen formulas that together proved step 15 of the 18-step proof. Nothing new of interest appeared, although I did find slightly different proofs of the steps that I had already proved. I then adjoined those so-called extra proof steps, in the spirit of further cramming. Although I found still newer proofs of steps already proved, no progress was occurring. What could I do? I decided to focus on the second proof—that which is bidirectional and relies on demodulation—as well as the first. While neither was what I wanted, I thought that perhaps using them as guidelines would help me reach the goal.

To see what challenges are presented by the second proof, I now offer it.

A Proof That Is Bidirectional and That Relies on Demodulation

6 [] x*rs(x,y) = y.

7 [] rs(x,x*y) = y.
8 [] ls(x,y)*y = x.
9 [] ls(x*y,y) = x.
10 [] 1*x = x.
11 [] x*1 = x.
12 [] x = x.
13 [] x*rs(x,y) = y.
16,15 [] rs(x,x*y) = y.
17 [] ls(x,y)*y = x.
19 [] ls(x*y,y) = x.
22,21 [] 1*x = x.
24,23 [] x*1 = x.
25 [] (x*y)* (z*x) = (x* (y*z))*x.
26 [copy,25,flip.1] (x* (y*z))*x = (x*y)* (z*x).
28 [] ((a*b)*c)*b != a* (b* (c*b)) | $ANS(m2).
30,29 [para_into,26.1.1.1.2,21.1.1,demod,24] (x*y)*x = x* (y*x).
31 [para_into,26.1.1.1.2,17.1.1,demod,30,flip.1] (x*ls(y,z))* (z*x) = x* (y*x).
33 [para_into,26.1.1.1.2,13.1.1,demod,30,flip.1] (x*y)* (rs(y,z)*x) = x* (z*x).
37 [para_into,26.1.2.1,13.1.1,demod,30] x* ((rs(x,y)*z)*x) = y* (z*x).
43 (heat=1) [para_into,29.1.1.1,6.1.1,flip.1] x* (rs(x,y)*x) = y*x.
47,46 (heat=1) [para_from,29.1.1,9.1.1.1] ls(x* (y*x),x) = x*y.
70,69 (heat=1) [para_from,33.1.1,7.1.1.2] rs(x*y,x* (z*x)) = rs(y,z)*x.
113 [para_from,26.1.1,15.1.1.2] rs(x* (y*z),(x*y)* (z*x)) = x.
126,125 (heat=1) [para_into,113.1.1.2.1,11.1.1,demod,70,22] rs(x,x)*y = y.
136,135 [para_into,125.1.1,23.1.1] rs(x,x) = 1.
176 [para_from,43.1.1,15.1.1.2] rs(x,y*x) = rs(x,y)*x.
204 (heat=1) [para_into,176.1.1.2,10.1.1,demod,136,flip.1] rs(x,1)*x = 1.
209 [para_from,204.1.1,26.1.2.2,demod,30,24] x* ((y*rs(x,1))*x) = x*y.
213 [para_from,204.1.1,26.1.2.1,demod,30,22] rs(x,1)* ((x*y)*rs(x,1)) = y*rs(x,1).
216,215 [para_from,204.1.1,19.1.1.1] ls(1,x) = rs(x,1).
223 (heat=1) [para_from,209.1.1,7.1.1.2,demod,16,flip.1] (x*rs(y,1))*y = x.
244 (heat=1) [para_from,213.1.2,9.1.1.1,demod,47] rs(x,1)* (x*y) = y.
246 [para_from,31.1.1,15.1.1.2,demod,70] rs(ls(x,y),x)*z = y*z.
248 (heat=1) [para_into,246.1.1,11.1.1,demod,24] rs(ls(x,y),x) = y.
258 [para_into,223.1.1.1.2,248.1.1,demod,216] (x*y)*rs(y,1) = x.
267,266 (heat=1) [para_into,258.1.1.1,8.1.1,flip.1] ls(x,y) = x*rs(y,1).
268 (heat=1) [para_into,258.1.1.1,6.1.1] x*rs(rs(y,x),1) = y.
312 [para_into,244.1.1.1,248.1.1,demod,267,22] x* (rs(x,1)*y) = y.
326 (heat=1) [para_into,312.1.1.2,6.1.1,flip.1] rs(rs(x,1),y) = x*y.
523 [para_from,268.1.1,37.1.1.2.1,flip.1] x* (rs(rs(y,rs(z,x)),1)*z) = z* (y*z).
528,527 [para_from,268.1.1,244.1.1.2,flip.1] rs(rs(x,y),1) = rs(y,1)*x.
530,529 (heat=1) [para_from,523.1.1,7.1.1.2,demod,528,528] rs(x,y* (z*y)) = ((rs(x,1)*y)*z)*y.
615 [para_from,326.1.2,28.1.2,demod,530,528,126] ((a*b)*c)*b != ((a*b)*c)*b | $ANS(m2).
616 [binary,615.1,12.1] $ANS(m2).

Sixteen of the steps of the proof rely on demodulation, and the last is the only step in which != occurs. Unfortunately, the proof completes by finding a unit conflict with reflexivity, $x = x$.

My plan was to take the first twenty-seven steps, all positive, and seek to prove as many of them as I could. Then, as is so evident in this notebook, iteration would come into play, one run depending on results from earlier runs. In subsequent experiments, therefore, I included negations of the first twenty-seven steps of the proof, hoping I could find a means to finish things off in the context of the last step, that which conflicts with reflexivity. I did not stop to worry about the fact that fifteen of the twenty-seven intermediate targets relied on demodulation. That fact just contributed to the labyrinthian nature of the expedition.

Cramming was my choice; indeed, in the beginning of this phase, I had in list(sos) Moufang 1, of course, followed by fifteen equations that form a proof of step fifteen of the 18-step flawed proof, followed by nine equations obtained by cramming on the fifteen. All started very fast: The experiment proved twenty-three of the twenty-seven targets, seven of which had already been (in effect) proved. On the twenty-seven to prove, all that remained were steps 21, 24, 26, and 27. I still kept in mind the possibly overwhelming obstacle that I would encounter when and if all twenty-seven had been proved, namely, how to travel that last bit to the proof of Moufang 2, how to cope with the small amount of bidirectionality and the demodulation that occurs at the end of the given proof. In particular, demodulation is used three times in the final deduction (in the proof I gave earlier). At the same time, in the context of the 18-step proof (which, if completed without a flaw, would more than suffice), all but steps 16 and 18 were proved. Some of those steps had a proof length of 0; after all, as noted, I had included a 15-step proof of step 15 in the initial set of support. I experienced no disappointment or frustration at this point, and, for the planned iteration, I took from the experiment thirty equations (after sorting to remove duplicates) to be used in further cramming.

In the next experiment, all was proceeding well. Indeed, the 24th and 26th steps of the twenty-seven to be proved (of the second proof based on demodulation) were proved. I extracted thirteen additional equations from the output of that run, equations to be used in yet more cramming. For clarity, I note that the two new proofs of so-called intermediate steps, 24 and 26, were each but two steps long. However, more new equations were found because different proofs of already-proved items were completed. Yes, if you tried to amass proofs of some item, you might be surprised at the large number of different proofs that exist and that can be found by an automated reasoning program. Some of them are most counterintuitive.

Efforts to prove the 27th step of the twenty-seven positive steps of the proof that is bidirectional and relies on demodulation failed repeatedly. Finally, I realized that perhaps I needed to interchange its arguments, as I have illustrated with earlier proofs of Moufang theorems. Therefore, I placed the negation of 27, calling it KEY, and the negation with the arguments interchanged, calling that KEYALT. What a fortuitous move: Still with cramming, using the many added equations already discussed, OTTER found a 2-step proof of KEYALT, of step 27 with the two major arguments of the equality interchanged. This trip is indeed circuitous. All that remained was to figure out a way of completing a proof of Moufang 2, of overcoming the fact that the last step of the second proof in focus relies on three applications of demodulation and is itself a not-equal item.

The light was beginning to shine, to appear brighter and brighter. Yes, I believe if the demodulation-based proof did not complete with the use of reflexivity, all would have been simpler. As it was, I conjectured that the following might produce enough fuel to complete the voyage. I took the following parent of the last step of the demodulation-based proof and submitted it to demodulation, using as demodulators the three used in the last step of the proof in focus.

```
326 (heat=1) [para_into,312.1.1.2,6.1.1,flip.1] rs(rs(x,1),y) = x*y.  %  parent to be demodulated
%  three demodulators, taken from the proof in focus
530,529 (heat=1) [para_from,523.1.1,7.1.1.2,demod,528,528] rs(x,y* (z*y)) = ((rs(x,1)*y)*z)*y.
528,527 [para_from,268.1.1,244.1.1.2,flip.1] rs(rs(x,y),1) = rs(y,1)*x.
126,125 (heat=1) [para_into,113.1.1.2.1,11.1.1,demod,70,22] rs(x,x)*y = y.
```

To obtain the results at each stage, I made a run with the command set(very_verbose), a command that has OTTER place in the output file copious detail. My plan was to extract various targets that, when proven, might enable the program to prove Moufang 2. In other words, I hoped to prove certain lemmas, each corresponding to something taken from the very verbose run, and then use these lemmas as hints rather than as added assumptions. If local success was OTTER's, then the proof steps along the way, along this tortuous voyage, would all be used as hints. (I chose hints over resonators because OTTER proceeds faster with hints than with resonators; however, sometimes resonators produce results that hints do not.) The four targets, gleaned from the very verbose run, in negated form are the following.

```
((a*b)*c)*b != rs(rs(a,1),b* (c*b)) | $ANS(INTER3).
rs(rs(a,1),b* (c*b)) != ((a*b)*c)*b | $ANS(INTER3).
((rs(rs(a,1),1)*b)*c)*b != ((a*b)*c)*b | $ANS(INTER3).
(((rs(1,1)*a)*b)*c)*b != ((a*b)*c)*b | $ANS(INTER3).
```

I submitted the following input file, hoping for total victory.

An Input File That Derives Moufang 2 from Moufang 1

```
% Sample Input File for the Study of Moufang Loops
op(400,xfx, *). % make all association explicit
% set(knuth_bendix).

set(para_from).
set(para_into).
set(order_eq).
set(hyper_res).
% set(ancestor_subsume).
set(back_sub).
% set(print_proof_as_hints).

assign(max_weight,23).
% assign(change_limit_after,100).
% assign(new_max_weight,12).
assign(equiv_hint_wt,1).
set(keep_hint_equivalents).
% set(sos_queue).
% set(process_input).

clear(print_kept).
clear(print_back_sub).
clear(print_new_demod).
clear(print_back_demod).

assign(max_proofs,-1).
assign(pick_given_ratio,2).
assign(max_distinct_vars,3).
assign(heat,0).
% assign(max_seconds,30).

weight_list(pick_given).
% including these from a draft of a Moufang paper, perhaps from 1995.
weight((x*y)*x=x* (y*x),0).
weight((x*ls(y,z))* (z*x)=x* (y*x),0).
weight((x*y)* (rs(y,z)*x)=x* (z*x),0).
weight(x* ((rs(x,y)*z)*x)=y* (z*x),0).
weight(x* (rs(x,y)*x)=y*x,0).
weight(ls(x* (y*x),x)=x*y,0).
weight(rs(x*y,x* (z*x))=rs(y,z)*x,0).
weight(rs(x* (y*z),(x*y)* (z*x))=x,0).
weight(rs(x,x)*y=y,0).
weight(rs(x,x)=1,0).
weight(rs(x,y*x)=rs(x,y)*x,0).
weight(rs(x,0)*x=1,1).
weight(x* ((y*rs(x,0))*x)=x*y,1).
weight(rs(x,0)* ((x*y)*rs(x,1))=y*rs(x,1),1).
weight(ls(1,x)=rs(x,0),1).
weight((x*rs(y,0))*y=x,1).
```

```
weight(rs(x,0)* (x*y)=y,1).
weight(rs(ls(x,y),x)*z=y*z,0).
weight(rs(ls(x,y),x)=y,0).
weight((x*y)*rs(y,0)=x,1).
weight(ls(x,y)=x*rs(y,0),1).
weight(x*rs(rs(y,x),0)=y,1).
weight(x* (rs(x,0)*y)=y,1).
weight(rs(rs(x,0),y)=x*y,1).
weight(x* (rs(rs(y,rs(z,x)),0)*z)=z* (y*z),1).
weight(rs(rs(x,y),0)=rs(y,1)*x,1).
weight(rs(x,y* (z*y))= ((rs(x,0)*y)*z)*y,1).
weight(((a*b)*c)*b!= ((a*b)*c)*b|$ANS(m2),0).
weight((x* (1*y))*x=x* (y*x),1).
weight((x*ls(y,z))* (z*x)= (x*y)*x,1).
weight((x*y)*x=x* (y*x),1).
weight(x* (rs(x,y)*x)=y*x,1).
weight((x*ls(y,z))* (z*x)=x* (y*x),1).
weight(1* (x*1)=1*x,1).
weight(rs(x,y*x)=rs(x,y)*x,1).
weight(x*1=1*x,1).
weight(rs(x,1*x)=1,1).
weight(rs(x,1)*x=1,1).
weight((x*ls(y,rs(x,1)))*1=x* (y*x),1).
weight(x*ls(y,rs(x,1))=x* (y*x),1).
weight(rs(x,x* (y*x))=ls(y,rs(x,1)),1).
weight(ls(x,rs(y,1))=x*y,1).
weight((x*rs(y,1))*y=x,1).
weight((x*rs(y,1))* (y* (z*y))= (x*z)*y,1).
weight((x*y)*rs(y,1)=x,1).
weight(((x*y)*z)*y=x* (y* (z*y)),1).
end_of_list.

%  The following list can be used to purge unwanted equations
weight_list(purge_gen).
%  Blocking use of Moufang 3.
weight((x * (y * (x * z)) = ((x * y) * x) * z), 1000).
weight((((x * y) * x) * z = x * (y * (x * z))), 1000).
%  Blocking use of Moufang 4.
weight((x * ((y * z) * x) = (x * y) * (z * x)), 1000).
weight(((x * y) * (z * x) = x * ((y * z) * x)), 1000).
end_of_list.

%  Used to complete applications of inference rules.
list(usable).
x = x.
x * rs(x,y) = y.  %  right solvable
rs(x, x * y) = y.  %  right solution is unique (implies left cancellation)
ls(x,y) * y = x.  %  left solvable
ls(x * y , y) = x.  %  left solution is unique (implies right cancellation)

%  identity:
1 * x = x .
x * 1 = x.
```

```
% left cancellation
% x*y != u | x*z != u | y = z.
% right cancellation
% y*x != u | z*x != u | y = z.
end_of_list.

% Used to initiate applications of inference rules.
list(sos).
% Axiom, Moufang 1:
(x * y) * (z * x) = (x * (y * z)) * x.
end_of_list.

% Used mainly to detect proof completion and to monitor progress.
list(passive).
((a*b)*c)*b != rs(rs(a,1),b* (c*b)) | $ANS(INTER3).
rs(rs(a,1),b* (c*b)) != ((a*b)*c)*b | $ANS(INTER3).
((rs(rs(a,1),1)*b)*c)*b != ((a*b)*c)*b | $ANS(INTER3).
(((rs(1,1)*a)*b)*c)*b != ((a*b)*c)*b | $ANS(INTER3).
a* ((rs(rs(b,a),1)*c)*b) != b* (c*b) | $ANS(INTER2).
a* (((rs(a,1)*b)*c)*b) != b* (c*b) | $ANS(INTER2).
((rs(a,1)*b)*c)*b != rs(a,b* (c*b)) | $ANS(INTER2).
rs(a,b* (c*b)) != ((rs(a,1)*b)*c)*b | $ANS(KEY).
((rs(a,1)*b)*c)*b != rs(a,b* (c*b))  | $ANS(KEYALT).
% Following 27 negs from a Knuth-Bendix proof
% (a*b)*a != a* (b*a) | $ANS(INTER1).
% (a*ls(b,c))* (c*a) != a* (b*a) | $ANS(INTER1).
% (a*b)* (rs(b,c)*a) != a* (c*a) | $ANS(INTER1).
% a* ((rs(a,b)*c)*a) != b* (c*a) | $ANS(INTER1).
% a* (rs(a,b)*a) != b*a | $ANS(INTER1).
% ls(a* (b*a),a) != a*b | $ANS(INTER1).
% rs(a*b,a* (c*a)) != rs(b,c)*a | $ANS(INTER1).
% rs(a* (b*c),(a*b)* (c*a)) != a | $ANS(INTER1).
% rs(a,a)*b != b | $ANS(INTER1).
% rs(a,a) != 1 | $ANS(INTER1).
% rs(a,b*a) != rs(a,b)*a | $ANS(INTER1).
% rs(a,1)*a != 1 | $ANS(INTER1).
% a* ((b*rs(a,1))*a) != a*b | $ANS(INTER1).
% rs(a,1)* ((a*b)*rs(a,1)) != b*rs(a,1) | $ANS(INTER1).
% ls(1,a) != rs(a,1) | $ANS(INTER1).
% (a*rs(b,1))*b != a | $ANS(INTER1).
% rs(a,1)* (a*b) != b | $ANS(INTER1).
% rs(ls(a,b),a)*c != b*c | $ANS(INTER1).
% rs(ls(a,b),a) != b | $ANS(INTER1).
% (a*b)*rs(b,1) != a | $ANS(INTER1).
% ls(a,b) != a*rs(b,1) | $ANS(INTER1).
% a*rs(rs(b,a),1) != b | $ANS(INTER1).
a* (rs(a,1)*b) != b | $ANS(INTER1).
rs(rs(a,1),b) != a*b | $ANS(INTER1).
a* (rs(rs(b,rs(c,a)),1)*c) != c* (b*c) | $ANS(INTER1).
rs(rs(a,b),1) != rs(b,1)*a | $ANS(INTER1).
rs(a,b* (c*b)) != ((rs(a,1)*b)*c)*b | $ANS(INTER1).
% Following negs for an 18-step proof, but hot list used illegally.
(a* (1*b))*a!=a* (b*a) | $ANS(INTER).
```

```
% (a*ls(b,c))* (c*a)!= (a*b)*a | $ANS(INTER).
% (a*b)*a!=a* (b*a) | $ANS(INTER).
% a* (rs(a,b)*a)!=b*a | $ANS(INTER).
% (a*ls(b,c))* (c*a)!=a* (b*a) | $ANS(INTER).
% 1* (a*1)!=1*a | $ANS(INTER).
% rs(a,b*a)!=rs(a,b)*a | $ANS(INTER).
% a*1!=1*a | $ANS(INTER).
% rs(a,1*a)!=1 | $ANS(INTER).
% rs(a,1)*a!=1 | $ANS(INTER).
% (a*ls(b,rs(a,1)))*1!=a* (b*a) | $ANS(INTER).
% a*ls(b,rs(a,1))!=a* (b*a) | $ANS(INTER).
% rs(a,a* (b*a))!=ls(b,rs(a,1)) | $ANS(INTER).
% ls(a,rs(b,1))!=a*b | $ANS(INTER).
(a*rs(b,1))*b!=a | $ANS(INTER).
(a*rs(b,1))* (b* (c*b))!= (a*c)*b | $ANS(INTER).
(a*b)*rs(b,1)!=a | $ANS(INTER).
((a*b)*c)*b!=a* (b* (c*b)) | $ANS(INTER).
% Negation Axiom, Moufang 2:
((a * b) * c) * b  != a * (b * (c * b))  |  $ANS(m2).
a * (b * (c * b)) != ((a * b) * c) * b  | $ANS(m2a).
end_of_list.

% The following list can be used to purge unwanted equations of various types.
list(demodulators).
% Blocking use of Moufang 1.
% EQ(((x * y) * (z * x) = (x * (y * z)) * x), $T).
% EQ(((x * (y * z)) * x = ((x * y) * (z * x))), $T).
% Blocking use of Moufang 2.
% EQ((((x * y) * z) * y = x * (y * (z * y))), $T).
% EQ(((x * (y * (z * y))) = (((x * y) * z) * y)), $T).
% % Blocking use of Moufang 3.
% EQ((x * (y * (x * z)) = ((x * y) * x) * z), $T).
% EQ((((x * y) * x) * z = x * (y * (x * z))), $T).
% % Blocking use of Moufang 4.
%  EQ((x * ((y * z) * x) = (x * y) * (z * x)), $T).
%  EQ(((x * y) * (z * x) = x * ((y * z) * x)), $T).
end_of_list.

list(hints).
% Following 15 prove step 15 of an 18-step proof.
(x* (1*y))*x=x* (y*x).
(x*y)*x=x* (y*x).
x* (rs(x,y)*x)=y*x.
(x*y)* (z*x)=x* ((y*z)*x).
1* (x*1)=1*x.
rs(x,y*x)=rs(x,y)*x.
x*1=1*x.
rs(x,1*x)=1.
rs(x,1)*x=1.
x* ((y*rs(x,1))*x)= (x*y)*1.
(x*ls(y,rs(x,1)))*1=x* (y*x).
x*ls(y,rs(x,1))=x* (y*x).
rs(x,x* (y*x))=ls(y,rs(x,1)).
```

ls(x,rs(y,1))=x*y.
(x*rs(y,1))*y=x.
% Following 9 sorted from cramming on preceding 15, emp.moufang.1imp2.new.out1b.
(1* (rs(1,x)*y))*1=x*y.
(1*rs(1,x))*y=x* (y*1).
(1* (x*y))*1= (1*x)*y.
rs(x,x* (y*x))*rs(x,1)=y.
(x*ls(y,z))* (z*x)= (x*y)*x.
(x*ls(y,z))* (z*x)=x* (y*x).
(x* (rs(x,y)*z))*x=y* (z*x).
x* (y*1)=x*y.
(x*y)*rs(y,1)=x.
% Following 30 sorted, from cramming on preceding, temp.moufang.1imp2.new.out1e.
ls((ls(x,y)* (y*z))*ls(x,y),z*ls(x,y))=x.
ls((x* (rs(x,y)*x))* (z*y),y)=y* (x*z).
ls(x*rs(y,1),rs(y,1))=rs(y,1)* (y*x).
ls(x* (y*x),x)=x*y.
(ls(x,y)* (y*z))*ls(x,y)=x* (z*ls(x,y)).
ls((x* (y*z))*x,z*x)=x*y.
rs(ls(x,y),x)=y.
rs(ls(x,y),x)*z=y*z.
rs(x,1)* ((x*y)*rs(x,1))=1* (y*rs(x,1)).
rs(x,1)* ((x*y)*rs(x,1))=y*rs(x,1).
rs(x,1)* (x*y)=y.
rs(x,ls((x* (y*z))*x,z*x))=y.
rs(x*ls(y,z),(x*y)*x)=z*x.
rs(x,x)=1.
rs(x,x)*y=y.
rs(x*y,(x* (y*z))*x)=z*x.
rs(x*y,(x*z)*x)=rs(y,z)*x.
rs(x*y,x* (z*x))=rs(y,z)*x.
rs(x* (y*z),(x*y)* (z*x))=x.
x*ls(y,rs(z,1))=x* (y*z).
x*rs(rs(y,x),1)=y.
x* (rs(x,rs(x,1))*x)=1.
(x* (rs(x,rs(x,1))*x))*y=y.
(x*rs(x,y))*ls(z,rs(x,1))=y* (z*x).
(x* (rs(x,y)*x))* (z*y)= (y* (x*z))*y.
x* ((rs(x,y)*z)*x)=y* (z*x).
(x*y)*ls(z,rs(x,1))=x* ((y*z)*x).
x* ((y*rs(x,1))*x)=x*y.
(x*y)* (rs(y,z)*x)= (x*z)*x.
(x*y)* (rs(y,z)*x)=x* (z*x).
% Following 13 prove more of the two proofs, the Knuth and the one with the hot-list-flaw.
ls(x*rs(y,1),rs(y,1))=rs(y,(y*x)*1).
rs(rs(x,1),ls(y*rs(x,1),rs(x,1)))=x*y.
rs(rs(x,1),y)=x*y.
rs(rs(x,y),1)=rs(y,1)*x.
rs(x,rs(y,1)*y)* (x*z)=z.
rs(x,(x*ls(y,rs(z,1)))*1)=y*z.
rs(x,(x*y)*1)=y.
rs(x,x* (y*z))=ls(y,rs(z,1)).
rs(x*y,1)*x=rs(y,1).

(x*ls(y,rs(z,1)))*1=x* (y*z).
x* (rs(rs(y,rs(z,x)),1)*z)=z* (y*z).
x* (rs(x,1)*y)=y.
x* ((y*rs(z,1))*z)=x*y.
% Following 2 from cramming on preceding.
x* (rs(rs(rs(x,y),1),z)*x)=y* (z*x).
x* ((rs(rs(y,x),1)*z)*y)=y* (z*y).
% Following 2 from cramming on preceding.
x* (rs(rs(y,1)*x,z)*x)=y* (z*x).
x* (((rs(x,1)*y)*z)*y)=y* (z*y).
% Following 2 from cramming on preceding
rs(x,y* (rs(rs(x,1)*y,z)*y))=z*y.
((rs(x,1)*y)*z)*y=rs(x,y* (z*y)).
% Following 2 proves second of the four new goals, temp.moufang.1imp2.new.out1r.
rs(rs(x,1),y* (z*y))= (((x*1)*y)*z)*y.
rs(rs(x,1),y* (z*y))= ((x*y)*z)*y.
% Following from cramming on preceding, temp.moufang.1imp2.new.out1r1.
((rs(rs(x,1),1)*y)*z)*y= ((x*y)*z)*y.
% Following 1 from cramming on preceding, temp.moufang.1imp2.new.out1r2.
(((rs(1,1)*x)*y)*z)*y= ((x*y)*z)*y.
% Following 149 borrowed from temp.moufang.3imp4.new.in1a
x*rs(x,y)=y.
y=x*rs(x,y).
rs(x,x*y)=y.
y=rs(x,x*y).
ls(x,y)*y=x.
x=ls(x,y)*y.
ls(x*y,y)=x.
x=ls(x*y,y).
1*x=x.
x=1*x.
x*1=x.
x=x*1.
x=x.
x=x.
x* (y* (x*z))= ((x*y)*x)*z.
((x*y)*x)*z=x* (y* (x*z)).
a* ((b*c)*a)!= (a*b)* (c*a).
(a*b)* (c*a)!=a* ((b*c)*a).
((x*rs(x,y))*x)*z=x* (rs(x,y)* (x*z)).
x* (rs(x,y)* (x*z))= ((x*rs(x,y))*x)*z.
(x*y)*z=y* (rs(y,x)* (y*z)).
y* (rs(y,x)* (y*z))= (x*y)*z.
x* (rs(x,y)* (x*z))= (y*x)*z.
(y*x)*z=x* (rs(x,y)* (x*z)).
((x*y)*x)*1= (x*y)*x.
(x*y)*x= ((x*y)*x)*1.
((x*y)*x)*1=x* (y* (x*1)).
x* (y* (x*1))= ((x*y)*x)*1.
(x*y)*x=x* (y* (x*1)).
x* (y* (x*1))= (x*y)*x.
(x*y)*x=x* (y*x).
x* (y*x)= (x*y)*x.

x* (rs(x,y)* (x*rs(x,z)))= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)* (x*rs(x,z))).
x* (rs(x,y)*z)= (y*x)*rs(x,z).
(y*x)*rs(x,z)=x* (rs(x,y)*z).
(x*y)*rs(y,z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)*rs(y,z).
(1*x)*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)= (1*x)*rs(x,y).
x*rs(x,y)=x* (rs(x,1)*y).
x* (rs(x,1)*y)=x*rs(x,y).
x=y* (rs(y,1)*x).
y* (rs(y,1)*x)=x.
x* (rs(x,1)*y)=y.
y=x* (rs(x,1)*y).
(ls(x,y)*y)*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)= (ls(x,y)*y)*rs(y,z).
x*rs(y,z)=y* (rs(y,ls(x,y))*z).
y* (rs(y,ls(x,y))*z)=x*rs(y,z).
(x*rs(x,y))*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))*rs(rs(x,y),z).
x*rs(rs(y,x),z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x*rs(rs(y,x),z).
ls((x*y)*rs(y,z),rs(y,z))=x*y.
x*y=ls((x*y)*rs(y,z),rs(y,z)).
ls(x* (rs(x,y)*z),rs(x,z))=y*x.
y*x=ls(x* (rs(x,y)*z),rs(x,z)).
x* (rs(x,ls(y,x))*z)=y*rs(x,z).
y*rs(x,z)=x* (rs(x,ls(y,x))*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y*rs(rs(x,y),z).
y*rs(rs(x,y),z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,ls(y,x))*1=rs(x,ls(y,x)).
rs(x,ls(y,x))=rs(x,ls(y,x))*1.
x* (rs(x,ls(y,x))*1)=y*rs(x,1).
y*rs(x,1)=x* (rs(x,ls(y,x))*1).
x*rs(x,ls(y,x))=y*rs(x,1).
y*rs(x,1)=x*rs(x,ls(y,x)).
x*rs(x,ls(y,x))=ls(y,x).
ls(y,x)=x*rs(x,ls(y,x)).
ls(x,y)=x*rs(y,1).
x*rs(y,1)=ls(x,y).
rs(rs(x,y),x)*1=rs(rs(x,y),x).
rs(rs(x,y),x)=rs(rs(x,y),x)*1.
rs(x,y)* (rs(rs(x,y),x)*1)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)* (rs(rs(x,y),x)*1).
rs(x,y)*rs(rs(x,y),x)=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=rs(x,y)*rs(rs(x,y),x).
rs(x,y)*rs(rs(x,y),x)=x.
x=rs(x,y)*rs(rs(x,y),x).
x=y*rs(rs(x,y),1).
y*rs(rs(x,y),1)=x.
x*rs(rs(y,x),1)=y.
y=x*rs(rs(y,x),1).
ls(x* (rs(x,y)*z),rs(x,z))= (x* (rs(x,y)*z))*rs(rs(x,z),1).

(x* (rs(x,y)*z))*rs(rs(x,z),1)=ls(x* (rs(x,y)*z),rs(x,z)).
(x* (rs(x,y)*z))*rs(rs(x,z),1)=y*x.
y*x= (x* (rs(x,y)*z))*rs(rs(x,z),1).
rs(x,y)*rs(rs(x,y),z)=z.
z=rs(x,y)*rs(rs(x,y),z).
(x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1)=y*x.
y*x= (x* (rs(x,y)*rs(rs(x,y),z)))*rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(x,rs(rs(x,z),y)),1)=z*x.
z*x= (x*y)*rs(rs(x,rs(rs(x,z),y)),1).
rs(x,x*rs(rs(y,x),1))=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,x*rs(rs(y,x),1)).
rs(x,y)=rs(rs(y,x),1).
rs(rs(y,x),1)=rs(x,y).
rs(rs(x,y),1)=rs(y,x).
rs(y,x)=rs(rs(x,y),1).
rs(rs(x,rs(rs(x,y),z)),1)=rs(rs(rs(x,y),z),x).
rs(rs(rs(x,y),z),x)=rs(rs(x,rs(rs(x,y),z)),1).
(x*y)*rs(rs(rs(x,z),y),x)=z*x.
z*x= (x*y)*rs(rs(rs(x,z),y),x).
(x*y)*rs(rs(rs(x,x*z),y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(rs(x,x*z),y),x).
(x*y)*rs(rs(z,y),x)= (x*z)*x.
(x*z)*x= (x*y)*rs(rs(z,y),x).
(x*y)*rs(rs(z,y),x)=x* (z*x).
x* (z*x)= (x*y)*rs(rs(z,y),x).
x* (rs(x,y)* (x* (rs(x,1)*z)))= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)* (x* (rs(x,1)*z))).
x* (rs(x,y)*z)= (y*x)* (rs(x,1)*z).
(y*x)* (rs(x,1)*z)=x* (rs(x,y)*z).
(x*y)* (rs(y,1)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)= (x*y)* (rs(y,1)*z).
(x*rs(x,y))* (rs(rs(x,y),1)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
rs(x,y)* (rs(rs(x,y),x)*z)= (x*rs(x,y))* (rs(rs(x,y),1)*z).
x* (rs(rs(y,x),1)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(rs(y,x),1)*z).
x* (rs(x,y)*z)=rs(y,x)* (rs(rs(y,x),y)*z).
rs(y,x)* (rs(rs(y,x),y)*z)=x* (rs(x,y)*z).
rs(x,y)* (rs(rs(x,y),x)*z)=y* (rs(y,x)*z).
y* (rs(y,x)*z)=rs(x,y)* (rs(rs(x,y),x)*z).
x* (rs(x,y)*z)=x*rs(rs(y,x),z).
x*rs(rs(y,x),z)=x* (rs(x,y)*z).
rs(x,x*rs(rs(y,x),z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x*rs(rs(y,x),z)).
rs(x,x* (rs(x,y)*z))=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,x* (rs(x,y)*z)).
rs(x,x* (rs(x,y)*z))=rs(x,y)*z.
rs(x,y)*z=rs(x,x* (rs(x,y)*z)).
rs(x,y)*z=rs(rs(y,x),z).
rs(rs(y,x),z)=rs(x,y)*z.
rs(rs(x,y),z)=rs(y,x)*z.
rs(y,x)*z=rs(rs(x,y),z).
(x*y)* (rs(y,z)*x)=x* (z*x).
x* (z*x)= (x*y)* (rs(y,z)*x).

```
(x*y)* (rs(y,y*z)*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (rs(y,y*z)*x).
(x*y)* (z*x)=x* ((y*z)*x).
x* ((y*z)*x)= (x*y)* (z*x).
a* ((b*c)*a)= (a*b)* (c*a).
(a*b)* (c*a)=a* ((b*c)*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)!= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
(a*b)* (c*a)= (a*b)* (c*a).
$F.
(x* (1*y))*x=x* (y*x).
(x*ls(y,z))* (z*x)= (x*y)*x.
(x*y)*x=x* (y*x).
(x*ls(1*y,z))* (z*x)=x* (y*x).
(x*ls(y,z))* (z*x)=x* (y*x).
x* (rs(x,y)*x)=y*x.
1* (x*1)=1*x.
rs(x,y*x)=rs(x,y)*x.
x*1=1*x.
rs(x,1*x)=1.
rs(x,1)*x=1.
(x*ls(y,rs(x,1)))*1=x* (y*x).
x*ls(y,rs(x,1))=x* (y*x).
rs(x,x* (y*x))=ls(y,rs(x,1)).
ls(x,rs(y,1))=x*y.
(x*rs(y,1))*y=x.
(x*rs(y,1))* (y* (z*y))= (x*z)*y.
(x*y)*rs(y,1)=x.
((x*y)*z)*y=x* (y* (z*y)).
end_of_list.

% Used for the hot list strategy.
list(hot).
% Axiom, Moufang 1:
(x * y) * (z * x) = (x * (y * z)) * x.
x * rs(x,y) = y.  % right solvable
rs(x, x * y) = y.  % right solution is unique (implies left cancellation)
ls(x,y) * y = x.  % left solvable
ls(x * y , y) = x.  % left solution is unique (implies right cancellation)
% identity:
1 * x = x .
x * 1 = x.
end_of_list.
```

A Proof of Moufang 2 from Moufang 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Sat Aug  1 17:06:00 2009
The command was "otter".  The process ID is 16037.
```

----> UNIT CONFLICT at 5.28 sec ----> 18966 [binary,18965.1,28.1] $ANS(m2).

Length of proof is 35. Level of proof is 18.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
4 [] ls(x,y)*y=x.
5 [] ls(x*y,y)=x.
6 [] 1*x=x.
7 [] x*1=x.
8 [] (x*y)* (z*x)= (x* (y*z))*x.
28 [] ((a*b)*c)*b!=a* (b* (c*b))|$ANS(m2).
282 [para_into,8.1.1.1,7.1.1,flip.1] (x* (1*y))*x=x* (y*x).
286 [para_into,8.1.1.1,2.1.1,flip.1] (x* (rs(x,y)*z))*x=y* (z*x).
293 [para_into,8.1.2.1.2,4.1.1] (x*ls(y,z))* (z*x)= (x*y)*x.
301 [para_from,8.1.2,5.1.1.1] ls((x*y)* (z*x),x)=x* (y*z).
305 [para_into,282.1.1.1.2,6.1.1] (x*y)*x=x* (y*x).
397 [para_into,286.1.2.2,7.1.1] (1* (rs(1,x)*y))*1=x*y.
782 [para_into,305.1.1.1,2.1.1,flip.1] x* (rs(x,y)*x)=y*x.
789 [para_into,305.1.1,293.1.2] (x*ls(y,z))* (z*x)=x* (y*x).
794 [para_into,305.1.1,286.1.1,flip.1] x* ((rs(x,y)*z)*x)=y* (z*x).
1014 [para_into,397.1.1,286.1.1] x* (y*1)=x*y.
1847 [para_from,782.1.1,3.1.1.2] rs(x,y*x)=rs(x,y)*x.
2687 [para_into,301.1.1.1.1,782.1.2] ls((x* (rs(x,y)*x))* (z*y),y)=y* (x*z).
3455 [para_into,1014.1.1,6.1.1] x*1=1*x.
5270 [para_from,3455.1.1,3.1.1.2] rs(x,1*x)=1.
5745 [para_into,5270.1.1,1847.1.1] rs(x,1)*x=1.
5934 [para_into,5745.1.1,782.1.2] x* (rs(x,rs(x,1))*x)=1.
5967 [para_from,5745.1.1,789.1.1.2] (x*ls(y,rs(x,1)))*1=x* (y*x).
6205 [para_from,5934.1.2,6.1.1.1] (x* (rs(x,rs(x,1))*x))*y=y.
6552 [para_into,5967.1.1,7.1.1] x*ls(y,rs(x,1))=x* (y*x).
7143 [para_from,6205.1.1,2687.1.1.1] ls(x*rs(y,1),rs(y,1))=rs(y,1)* (y*x).
7908 [para_from,6552.1.1,3.1.1.2] rs(x,x* (y*x))=ls(y,rs(x,1)).
8211 [para_into,7143.1.1,5.1.1,flip.1] rs(x,1)* (x*y)=y.
8667 [para_into,7908.1.1,3.1.1,flip.1] ls(x,rs(y,1))=x*y.
8690 [para_into,8211.1.1.1.2,5745.1.2] rs(x,rs(y,1)*y)* (x*z)=z.
8902 [para_from,8211.1.1,3.1.1.2] rs(rs(x,1),y)=x*y.
9412 [para_from,8667.1.1,4.1.1.1] (x*y)*rs(y,1)=x.
10029 [para_into,8690.1.1.1,3.1.1] x* (rs(x,1)*y)=y.
11227 [para_into,9412.1.1.1,2.1.1] x*rs(rs(y,x),1)=y.
11847 [para_from,10029.1.1,794.1.1.2.1,flip.1] x* ((rs(rs(y,x),1)*z)*y)=y* (z*y).
12875 [para_from,11227.1.1,8211.1.1.2,flip.1] rs(rs(x,y),1)=rs(y,1)*x.
14383 [para_from,12875.1.1,11847.1.1.2.1.1] x* (((rs(x,1)*y)*z)*y)=y* (z*y).
15707 [para_from,14383.1.1,3.1.1.2,flip.1] ((rs(x,1)*y)*z)*y=rs(x,y* (z*y)).
17043 [para_into,15707.1.1.1.1.1,8902.1.1,flip.1] rs(rs(x,1),y* (z*y))= (((x*1)*y)*z)*y.
18480 [para_into,17043.1.2.1.1.1,7.1.1] rs(rs(x,1),y* (z*y))= ((x*y)*z)*y.
18965 [para_into,18480.1.1,8902.1.1,flip.1] ((x*y)*z)*y=x* (y* (z*y)).

An inspection of the just-given proof showed me that indeed one of the INTER3 goals was proved within the 35-step proof and that KEYALT was also proved. In other words, the harbor was reached, reached in part by using very verbose to find targets whose proofs were obtained in some runs and then used in later runs. These targets arose mainly from the desire to circumvent the use, in the demodulation-based, bidirectional proof, of demodulation and of its last clause. Put a slightly different way, the entire trip was successful in its goal of finding a proof free of demodulation and relying, in the context of its deduced steps, on positive clauses only. You might enjoy a rather thorough study of the given 35-step proof in the context of the earlier proof (that played such a key role) that depends on using demodulation and, also, in the context of the intermediate goals that marked the lengthy voyage just completed. Such a study will reveal, among other things, the roles that were played by the items met during the long trip, items such as that called KEYALT and at least one of the items called INTER3. Thus the length, 35, is accurate.

Of course, I then turned to my usual study of proof shortening, and OTTER eventually, after various experiments, returned the following proof.

A 22-Step Proof Deriving Moufang 2 from Moufang 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on crush.mcs.anl.gov,
Sat Jul 25 10:29:37 2009
The command was "otter". The process ID is 29782.

----> UNIT CONFLICT at   2.17 sec ----> 10941 [binary,10940.1,28.1] $ANS(m2).

Length of proof is 22. Level of proof is 9.

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

2 [] x*rs(x,y)=y.
3 [] rs(x,x*y)=y.
5 [] ls(x*y,y)=x.
6 [] 1*x=x.
7 [] x*1=x.
8 [] (x*y)* (z*x)= (x* (y*z))*x.
28 [] ((a*b)*c)*b!=a* (b* (c*b))|$ANS(m2).
282 [para_into,8.1.1.1,7.1.1,flip.1] (x* (1*y))*x=x* (y*x).
286 [para_into,8.1.1.1,2.1.1,flip.1] (x* (rs(x,y)*z))*x=y* (z*x).
294 [para_into,8.1.2.1.2,2.1.1] (x*y)* (rs(y,z)*x)= (x*z)*x.
299 [para_from,8.1.1,5.1.1.1] ls((x* (y*z))*x,z*x)=x*y.
305 [para_into,282.1.1.1.2,6.1.1] (x*y)*x=x* (y*x).
566 [para_from,294.1.1,3.1.1.2] rs(x*y,(x*z)*x)=rs(y,z)*x.
671 [para_from,299.1.2,3.1.1.2] rs(x,ls((x* (y*z))*x,z*x))=y.
794 [para_into,305.1.1,286.1.1,flip.1] x* ((rs(x,y)*z)*x)=y* (z*x).
1522 [para_into,566.1.1.2,305.1.1] rs(x*y,x* (z*x))=rs(y,z)*x.
1532 [para_into,566.1.1,3.1.1,flip.1] rs(x,x)*y=y.
1705 [para_into,671.1.1.2.1.1,286.1.2] rs(x,ls(((y* (rs(y,x)*z))*y)*x,y*x))=z.
1710 [para_into,671.1.1.2.1.1,6.1.1] rs(1,ls((x*y)*1,y*1))=x.
1866 [para_into,1710.1.1,1705.1.1,flip.1] x* (rs(x,1)*y)=y.
4767 [para_into,1532.1.1,7.1.1] rs(x,x)=1.
5087 [para_into,1866.1.1.2,2.1.1,flip.1] rs(rs(x,1),y)=x*y.
5143 [para_from,1866.1.1,794.1.1.2.1,flip.1] x* ((rs(rs(y,x),1)*z)*y)=y* (z*y).
6541 [para_into,4767.1.1,1522.1.1] rs(x*y,x)*y=1.
7006 [para_from,6541.1.1,3.1.1.2] rs(rs(x*y,x),1)=y.
7575 [para_into,7006.1.1.1.1,2.1.1] rs(rs(x,y),1)=rs(y,x).
8941 [para_from,5143.1.1,1866.1.1.2,flip.1] (rs(rs(x,rs(y,1)),1)*z)*x=y* (x* (z*x)).
10754 [para_into,7575.1.2,5087.1.1] rs(rs(x,rs(y,1)),1)=y*x.
```

10940 [para_from,10754.1.1,8941.1.1.1.1] ((x*y)*z)*y=x* (y* (z*y)).

4. Highlights, Review, and Random Notes

I wrote this notebook in part because Overbeek thought algebra was in order, in contrast to the various areas of logic covered in earlier notebooks. If all goes well, in still later notebooks, I shall turn to other areas of algebra that may include group theory, ring theory, quasilattices, and the like. A reading of the material offered here may enable you to share in my delight at answering open questions, meeting challenges, and finding proofs of interest to others. One of the more piquant and satisfying experiences that can occur is the discovery that lemmas thought to be essential to the proving of some particular theorem are, in fact, not needed at all; such has occurred in my research more than once. With regard to the concept of pure proofs featured in the discussion of Moufang loops, purity can be studied in various other areas, as I have in various areas of logic that include *C*4 and *C*5. For now, I again note that I did not answer the most obvious "why" question. Specifically, I have not as yet identified why I was able to make the added progress (here in 2009) in the context of finding more elegant proofs, when compared with my research of 2002 and earlier. I have, however, provided you with an array of approaches for attacking similar problems, in the context of various measures of elegance. That array includes the following, with the comment that, although the illustrations found here are in terms of OTTER notation and the like, the items are indeed general, not depending on McCune's program.

One of the first areas of choice focuses on the retention of newly deduced information. You wish to give your program enough latitude to permit it to complete the given assignment, and yet prevent it from drowning in new items or getting lost in the possibly huge forest of new conclusions. Therefore, experience, conjectures, and, yes, guesses are in order regarding the actions to take. OTTER offers max_weight, to which you assign a value that, for newly deduced items, has the program discard such if and when their weight (priority) exceeds the value assigned to max_weight. The weight, or priority or complexity, is ordinarily (with McCune's program) measured in terms of symbol count, not counting parentheses and commas. The lower the weight, the higher the priority for being chosen to initiate an application of an inference rule in use. You can weight entire equations or formulas, or ou can weight parts of such, as illustrated in the various input files given in this notebook. You might find it instructive to examine an output file to see for yourself the weights that are given to the various retained items. The other important action you can take, that I also take, regards max_distinct_vars, to which you assign a value. When a newly deduced item relies on a number of distinct variables that exceeds the value assigned to max_distinct_vars, the item is immediately discarded. In addition to aiding the program in retaining enough new information, but not too much, the latter parameter is used when seeking a proof of greater elegance than in hand, where elegance is measured (in this case) in terms of the maximum number of distinct variables present in a deduced item present in the sought-after proof. You have examples of such earlier in this notebook.

You have just been given a small review of how you can use restriction strategies, restricting the reasoning by having the program discard new information you say is of no current interest. Weight templates can also be used to direct the reasoning, as shown with resonators and with hints. I assert without reservation that both restriction strategies and direction strategies are crucial if the goal is to answer an open question, to meet a significant challenge, to find a first proof of a difficult theorem, or to find a more elegant proof than currently in hand.

Next is demodulation, a procedure formulated in the mid-1960s to enable an automated reasoning program to simplify and canonicalize. However, in this notebook, as well as in so much of my research, I use demodulation as what might be termed a local restriction strategy. Specifically, I use it to bock the retention of various items by demodulating the unwanted to junk. You see examples of that usage in the given input files. The use that dominates much of what you find here is in the context of proof shortening. Typically, I have the program focus on a proof in hand and, one at a time, block its steps when and if deduced, with the objective of finding a shorter proof. Sometimes, how wondrous, the new shorter proof is a subproof of the one in focus, a subproof in the sense that all of its deduced steps are among those of the proof that motivated the study. In particular, a given equation or formula can be deduced in many, many ways; in other words, an inspection of the shorter proof can reveal that some of its deduced steps now have a different set of parents. I rely on a program called otter-loop, written by McCune, to enable me to run (in one experiment) a possibly large number of jobs, each differing from the others in its focus on blocking a chosen deduced step of the proof in hand. For example, if I have a 300-step proof, I have otter-loop in one experiment run three hundred separate jobs, blocking the first deduced step, the second, and so on to the three hundredth. If one of those subexperiments indicates progress, I adjoin the corresponding demodulator and rerun the set of three hundred jobs. But, you naturally ask, can I give some general rules for deciding how to choose, for example, between two moves, each of which promises to yield a shorter proof than that in hand? Further, in that you may suspect the situation is complex, can I make a number of possibly relevant observations? The following remarks are based on decades of experimentation.

Often I find that the blocking (with demodulation or with weighting) of steps *j* and *k* will lead to a proof, say, one step shorter than that in hand. Although I could simply add both demodulators (or weight templates), I typically adjoin that which corresponds to the step that occurs later in the proof in hand. My guess, or conjecture, is that blocking the later step gives the program more time to replace its use. When the iteration (sequence of experiments) offers earlier and earlier steps to block, then very often you are nearing the progress that can be made along the line of attack. Sometimes, however, (in the experimentation) the step suggested (to block) is earlier than the preceding one was suggested (to block), then a later one, then a still later, then an earlier, and that (sequence is encouraging. Of course, you could have blocked steps two at a time and found a pair that yielded the progress (in proof length) that would occur after blocking the corresponding steps one at a time; but, if the proof has substantial length (undefined), the number of pairs to consider, and hence the number of runs, may indeed be prohibitive.

Next in focus is the level shown for the blocking of a step. Often, as you block one step after another, finding shorter and still shorter proofs, the level remains constant. In general, such is encouraging. When the level begins to increase, you are likely to be reaching the end of that part of the journey. The level almost never decreases, from what I can recall. The best situation, when the goal is to find shorter and still shorter proofs, occurs when the distance between the level and length of a proof in hand is larger rather than smaller. As an extreme example, when the level equals the length, you are probably in trouble. At the other end of the spectrum, when I was studying intuitionism, at one point, I had a proof of length 300 and level 20; much, much later in that study, I was able to find a proof of length 196 and level 22. On the other hand, although for me counterintuitive, if two runs in succession suggest that the same length proof will result if and when the corresponding two demodulators are adjoined, once in a while the second adjunction does yield a further advance.

In addition to the actions cited in this section, you can invoke the use of the hot list. In that list, you place the items you wish the program to visit each time a new item has been retained. Moreover, you are always given the choice between a breadth-first (level-saturation) search and that in which the complexity of retained information is taken into account. Although in the early 1980s I considered level saturation of little use—an amusing stance in hindsight—I now rely on it heavily when I use the cramming strategy. Cramming is yet one more approach that has led to finding a more elegant proof than the one in hand. Indeed, by forcing (or cramming) steps of some subproof of an intermediate step or of a member of a conjunction into a so-called total proof, you sometimes find a new and more satisfying total proof, even though the subproofs now present are longer than they were in the total proof that initiated the cramming.

Of course, as you now know, sometimes the sought-after elegance focuses on the avoidance of various lemmas; indeed, I have examples of such that succeed in a grand fashion, relying on approaches presented in this notebook. I often, as seen in this notebook, rely on lemma adjunction, a procedure that adjoins (usually) to the list(sos) proof steps from earlier experiments. When I do, I often turn to a level-saturation (breadth-first) search, noting that the newly adjoined elements (equations or formulas) are now at level 0 (input). Many of such adjunctions, in the preceding run, had high clause number, which made them sometimes almost out of reach to be keyed upon; in contrast, they are now keyed upon quickly.

As also is evident in this and my other notebooks, iteration is key to much of my research. I find that one run (typically) benefits from the results of an earlier run.

As for quantitative data, data that surprised me even after these many decades, in an experiment focusing on intuitionistic logic, before the assignment was completed, almost half a billion clauses were generated. During that study, the proof I sought was found upon retention of clause (3394459). Many years ago, a researcher in automated reasoning was heard to say that, although the Argonne group had indeed contributed many interesting problems for study, large numbers of the type just cited showed that the various problems were not being attacked appropriately. After all, logicians such as Lukasiewicz would not have considered so many new items. Well, yes, no person would; but that is in fact one of the advantages of using a program such as OTTER. The cited large numbers, to me, suggest that the task was most difficult. In a still later experiment in intuitionism, the desired proof was found after 700,000,000 formulas were generated, 3,000,000 were kept, and almost 500,000 CPU-seconds were required, all on a very fast computer.

I look forward to your contacting me by e-mail with proofs that you wish shortened or made more elegant in some specified manner. Ideally, notation reminiscent of that used by OTTER would be most welcome. Until the next notebook, I bid you adieu.