Uniform Strategies: The CADE-11 Theorem Proving Contest[1]

EWING L. LUSK and WILLIAM W. MCCUNE
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 60439-4844, U.S.A.
E-mail: {lusk,mccune}@mcs.anl.gov.

Abstract. At CADE-10 Ross Overbeek proposed a two-part contest to stimulate and reward work in automated theorem proving. The first part consists of seven theorems to be proved with resolution, and the second part of equational theorems. Our theorem provers OTTER and its parallel child ROO proved all of the resolution theorems and half of the equational theorems. This paper represents a family of entries in the contest.

Key Words. Automated theorem proving, resolution, paramodulation, Knuth-Bendix completion, strategy.

1 Introduction

The Conference on Automated Deduction (CADE) has been for nearly twenty years a meeting where both theoreticians and practitioners present their work. Feeling perhaps that the conference was becoming dominated by the theoreticians, Ross Overbeek proposed at CADE-10 in 1990 a contest to stimulate work on the implementation and use of automated theorem-proving systems. The challenge was to prove a set of theorems, and do so with a uniform approach. That is, one was not allowed to set parameters in the system to specialize it for individual problems. There were actually two separate contests, one represented by a set of seven problems designed to test basic inference components, and the other represented by a set of ten problems designed to test equality-based systems.

This paper describes the results of our preparation to enter the contest with OTTER [6, 7] and ROO [1, 2], two systems developed at Argonne National Laboratory. ROO is a parallel version of OTTER but has such different behavior in some cases that we treat them as separate entries. We entered each of them in both contests. Both programs proved all of the theorems in the basic set and the first five theorems in the equality set.

Some of the problems are difficult; and although many of the problems had been done before with OTTER, in each case we had set OTTER's many input parameters in a way customized to the problem at hand, and we had chosen a set of support that appeared to us to be most natural. It was a challenge to come up with a uniform set of parameter settings and a uniform algorithm for picking the set of support that would allow OTTER to prove the theorems. Indeed OTTER has sometimes been criticized for presenting the user with too many choices. The settings described here answer that criticism by providing a collection of parameter settings and a uniform algorithm for picking the set of support that can be used as a "default" set of choices. This relatively diverse set of problems can be used a starting point for further exploration.

[1] This work was supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

2 The Theorems

In this section we give informal statements of the theorems. The exact inputs to OTTER 2.2 are given in Appendices A and B.

2.1 The Basic Theorems

Theorems 1, 2, and 3 below, although normally thought of as equational theorems, are stated in the P-formulation, in which a ternary predictate, say $P(a,b,c)$, is an alternative representation of $a*b=c$. Equality is still required and is indeed quite useful for demodulation. Although not explicitly stated in the contest rules, we believe that demodulation is permitted, but that paramodulation is not. This has been the traditional approach when using the P-formulation [5, 4], and we have followed it.

Theorem 1. $x^2 = e$ groups are commutative, stated in the P-formulation.

Theorem 2. The commutator theorem: if a group has the property $x^3 = e$, then $[[x,y],y] = e$, where $[x,y] = xyx^{-1}y^{-1}$. The theorem is stated in the P-formulation.

Theorem 3. $x^2 = x$ rings are commutative, stated in the P-formulation.

Theorem 4. XGK is a single axiom for the equivalential calculus. More precisely, let XGK=$e(x,e(e(y,e(z,x)),e(z,y)))$ and PYO=$e(e(e(x,e(y,z)),z),e(y,x))$; then XGK $\rightarrow$ PYO by condensed detachment.

Theorem 5. The formula $i(i(i(x,y),z),i(i(z,x),i(u,x)))$ implies the formula $i(i(x,y),i(i(y,z),i(x,z)))$ by condensed detachment. We refer to this theorem as Imp-4, and it is CD-67 of the problem set in [9].

Theorem 6. The formula $i(i(x,y),i(i(z,x),i(z,y)))$ is a theorem in the many-valued sentential calculus. This is theorem CD-57 in [9].

Theorem 7. The formula $i(i(x,y),i(n(y),n(x)))$ is a theorem in the many-valued sentential calculus. This is theorem CD-60 in [9].

2.2 The Equality Theorems

Theorem EQ-1. The commutator theorem (see Theorem 2 above) in the equality formulation.

Theorem EQ-2. If a Robbins algebra has the property $(\exists c, c+c=c)$, then it is also a Boolean algebra.

Theorem EQ-3. Consider the 5 standard axioms for ternary Boolean algebra. If axiom 3 is removed, then $f(x,g(x),y) = y$ still holds.

Theorem EQ-4. The equality $f(x,i(f(f(i(f(i(y),f(i(x),w))),z),i(f(y,z))))) = w$ implies that f is associative. (The equality is, in fact, a single axiom for group theory.)

Theorem EQ-5. The equality $i(i(i(x,y),i(y,x)),i(y,x)) = T$ holds in an equational formulation of the many-valued sentential calculus. This is the Wajsberg algebra formulation of a conjecture of Łukasiewicz.

Theorem EQ-6. The fragment {B,W,M} of combinatory logic contains fixed point combinators.

Theorem EQ-7. Rings in which $x^3 = x$ are commutative.

Theorem EQ-8. The fragment {B,W} of combinatory logic contains fixed point combinators.

Theorems EQ-9. On three Moufang identities and nonassociative rings. See [11] for a statement of the theorem.

Theorems EQ-10. On right alternative nonassociative rings. As far as we know, this is still an open problem. See [11] for a statement of the theorem.

3 Results

OTTER and ROO proved all seven theorems in the basic set and the first five of the ten problems in the equality set. See Section 4 for the options and set of support used.

Tables 1 and 2 list the results for the two sets with OTTER, with ROO running on 8 processors, and with ROO running on 12 processors.

The OTTER jobs were run on a SPARCstation 2. We used OTTER 2.2, the version that was released in July 1991. The ROO jobs were run on an Alliant 2800 with 12 (Intel i860) processors. The version of ROO we used is based on OTTER 2.2xa+ (June 1992).

4 Settings and Set of Support

Within each set, all of the OTTER jobs used the same settings. However, the settings for the basic set were substantially different from those for the equality set. The ROO jobs used settings slightly different from the OTTER jobs, and (for small technical reasons) the ROO settings for the basic set varied slightly, depending on whether equality is present.

For the basic set, the initial set of support consisted of the positive input clauses, except $x = x$. For the equality set, the initial set of support depended on whether the theorem has an obvious special hypothesis: if so, then the set of support was the special hypothesis and the denial of the conclusion; if not, the set of support consisted of all input clauses except $x = x$.

The rules for the equality set state that an ordering on the symbols may be included with the input clauses. The ordering is typically used to orient equality literals. Our general rule for ordering symbols is "nonSkolem constants $\prec$ Skolem constants $\prec$ nondefined function symbols (by nonincreasing arity) $\prec$ defined function symbols". See Appendix B for the specific orderings that were used.

Our choices for the settings reflect long-term experience and a small amount of experimentation on the theorems of the contest.

Table 1: Results for Basic Theorems

	OTTER	ROO-8	ROO-12
Theorem 1: $x^2 = e$ Group			
proof time (sec.)	0.20	0.32	0.32
generated	222	2300	1867
kept	13	30	40
memory (K)	31	728	564
Theorem 2: Commutator			
proof time (sec.)	35.60	26.89	25.97
generated	20575	88838	131429
kept	4505	3684	1697
memory (K)	1564	12515	12670
Theorem 3: $x^2 = x$ Ring			
proof time (sec.)	145.41	35.57	38.18
generated	56025	134744	221890
kept	13990	4316	2736
memory (K)	4342	14333	18739
Theorem 4: XGK			
proof time (sec.)	407.50	159.87	55.37
generated	177109	663722	263233
kept	15320	16519	9466
memory (K)	8047	19539	22189
Theorem 5: Imp-4 (CD-67)			
proof time (sec.)	7711.98	1051.55	909.95
generated	8341570	7171447	8182376
kept	17862	14855	17666
memory (K)	10729	13983	15098
Theorem 6: MV-1 (CD-57)			
proof time (sec.)	17.68	4.37	14.71
generated	16687	24159	114051
kept	4837	1024	2000
memory (K)	2171	6479	12161
Theorem 7: MV-2 (CD-60)			
proof time (sec.)	2184.96	427.89	152.53
generated	3214280	4311090	1997084
kept	16250	12374	10750
memory (K)	7216	13664	13755

Table 2: Results for Equality Problems

	OTTER	ROO-8	ROO-12
Theorem EQ-1: Commutator			
proof time (sec.)	1.49	0.76	0.86
generated	542	1727	2144
kept	114	91	89
memory (K)	255	1208	1460
Theorem EQ-2: Robbins, $c + c = c$			
proof time (sec.)	98.19	18.63	13.43
generated	50001	56067	59151
kept	4548	2450	1235
memory (K)	5652	12676	13342
Theorem EQ-3: TBA			
proof time (sec.)	16.78	4.10	3.16
generated	3945	9307	11170
kept	1030	620	378
memory (K)	1564	4880	5043
Theorem EQ-4: Group single axiom			
proof time (sec.)	44.12	10.56	9.25
generated	3417	11778	16118
kept	2507	1015	863
memory (K)	4470	13889	17110
Theorem EQ-5: Wajsberg algebra			
proof time (sec.)	2248.86	425.99	491.67
generated	1012625	971543	1437272
kept	5897	4374	4022
memory (K)	6801	13376	14525

4.1 Settings for the Basic Set

OTTER: basic set	ROO: basic with equality	ROO: basic without equality
	set(index_for_back_demod)	
set(hyper_res)	set(hyper_res)	set(hyper_res)
set(back_demod)	set(back_demod)	
set(dynamic_demod_all)	set(dynamic_demod_all)	
assign(pick_given_ratio,5)	assign(pick_given_ratio,5)	assign(pick_given_ratio,5)
clear(print_kept)	clear(print_kept)	clear(print_kept)
assign(max_mem,20000)	assign(max_mem,32000)	assign(max_mem,32000)
set(control_memory)	set(control_memory)	set(control_memory)

4.2 Settings for the Equality Set

OTTER: equality set	ROO: equality set
set(knuth_bendix)	set(knuth_bendix)
set(index_for_back_demod)	set(index_for_back_demod)
set(process_input)	set(process_input)
assign(max_mem,16000)	assign(max_mem,32000)
set(control_memory)	set(control_memory)
set(lex_rpo)	set(lex_rpo)
clear(print_kept)	clear(print_kept)
clear(print_new_demod)	clear(print_new_demod)
clear(print_back_demod)	clear(print_back_demod)

4.3 Description of the Settings

set(hyper_res). This option activates the inference rule hyperresolution.

set(back_demod). When new equalities are deduced, this option causes them to be used as rewrite rules.

set(dynamic_demod_all). This option has OTTER use all new orientable equalities as rewrite rules.

assign(pick_given_ratio,5). By default OTTER chooses each new given clause based on its symbol count. Hence, a heavy clause that is needed for the proof cannot be used until all lighter clauses have been used. We have recently found it useful to mix this strategy with a breadth-first strategy by choosing some percentage of the given clauses according to the order in which they are generated rather than by weight. This particular setting causes the search to be five parts (i.e., given clauses) shortest-first to one part breadth-first.

clear(print_kept). clear(print_new_demod). clear(print_back_demod). These options suppress output, saving file space and a little time.

assign(max_mem,20000). This setting restricts memory usage to 20 megabytes. It is used in conjunction with the following parameter.

set(control_memory). This setting causes OTTER and ROO to periodically analyze memory usage and to automatically adjust the weight threshold of subsequent clauses if memory usage is high. The adjusted weight threshold is a function of the weight distribution of the clauses in the set of support list. See [9].

set(knuth_bendix). This option causes OTTER and ROO to automatically set a collection of options that approximate a Knuth-Bendix completion procedure. Under this option, the theorem prover orders equalities, paramodulates from left sides into left sides, and back demodulates.

set(index_for_back_demod). This option causes indexing to be used when searching for terms to which a new rewrite rule can be applied. ROO requires this "option" whenever back demodulation is enabled. OTTER frequently benefits from this option.

set(process_input). This option causes all input clauses to be processed (subsumption, demodulation, equality ordering, back demodulation) as if they were generated clauses.

set(lex_rpo). This option specifies the lexicographic recursive path ordering for comparing terms when attempting to orient equalities.

lex(*list of symbols***).** This command specifies an ordering on constant, function, and predicate symbols, with smallest first. For the experiments described in this paper, the ordering is used to attempt to orient equalities.

lrpo_lr_status(*list of symbols***).** This command specifies that function symbols are to be compared left to right when applying the lexicographic recursive path ordering.

5 Failures on Equality Theorems 6–10

Theorem EQ-6. *The fragment {B,W,M} of combinatory logic contains fixed point combinators.* This is not a difficult theorem, and OTTER has found a proof, but the settings were different from those used in theorems EQ-1 through EQ-5. The important difference was that the initial set of support consisted of the denial only (so that all generated clauses were negative), and paramodulation was allowed into both arguments of equality literals. The following input file causes OTTER 2.2 to find a proof of EQ-6 in about 27 seconds.

```
set(para_into).
clear(para_from_right).
set(order_eq).
assign(max_mem, 16000).
set(lex_rpo).
clear(print_kept).

lex([B,W,L,M,a(x,x),f(x)]).
lrpo_lr_status([a(x,x)]).

list(usable).
(x = x).
(a(a(a(B,x),y),z) = a(x,a(y,z))).
(a(a(W,x),y) = a(a(x,y),y)).
(a(M,x) = a(x,x)).
end_of_list.

list(sos).
(a(y,f(y)) != a(f(y),a(y,f(y)))) | $Ans(y).
end_of_list.
```

```
list(demodulators).
(a(a(a(B,x),y),z) = a(x,a(y,z))).
end_of_list.
```

Theorem EQ-7. *Rings in which $x^3 = x$ are commutative.* As far as we know, OTTER has never found a proof of this theorem, except with highly specialized settings and weight templates. (We have a new theorem prover, based on OTTER and with associative-commutative unification and matching, that can easily prove this theorem.)

Theorem EQ-8. *The fragment $\{B, W\}$ of combinatory logic contains fixed point combinators.* This theorem is much more difficult than EQ-6, and the strategy above that works for EQ-6 fails for EQ-8. The kernel method [8], which was developed for this type of problem, finds a proof of EQ-8 within a few seconds.

Theorems EQ-9 and EQ-10. On Moufang identities in nonassociative rings (EQ-9), and on right alternative nonassociative rings (EQ-10). The complicated definitions in these theorems cause terms in the conclusion to be greatly expanded. OTTER cannot cope with the complex conclusions, because it likes to focus on simple terms. As with (EQ-7), we believe that associative-commutative unification would be helpful for these theorems.

6 Conclusion

The ideal entry in the contest would have the program and its strategy fixed before attempting any of the theorems, but today's programs are simply not powerful or general enough to meet that challenge. We were able to devise nearly uniform strategies, which are not particularly complicated, for proving all of the theorems in the basic set and five of the ten theorems in the equality set. However, two parts of the strategy for the equality problems, set of support specification and symbol ordering, do involve some guidance from the user.

The program OTTER was not enhanced in any way for the contest; the final strategies were achieved with existing parameters. The program ROO is very experimental and much less predictable than OTTER; it required several small modifications to prove the theorems with uniform settings.

We thank Ross Overbeek for proposing this exercise, and we hope that others found it as useful as we did. The contest has taken us a bit further toward the goal of general-purpose, easy-to-use automated theorem provers.

Appendices

A Inputs to OTTER 2.2 for the Basic Set

A.1 Theorem 1: $x^2 = e$ Groups are Commutative (P-form)

```
set(hyper_res).
set(back_demod).
```

```
set(dynamic_demod_all).
assign(pick_given_ratio,5).
clear(print_kept).
assign(max_mem,20000).
set(control_memory).

list(usable).
-P(x,y,u) | -P(y,z,v) | -P(u,z,w) | P(x,v,w).
-P(x,y,u) | -P(y,z,v) | -P(x,v,w) | P(u,z,w).
-P(x,y,u) | -P(x,y,v) | eq(u,v).
eq(x,x).
-eq(x,y) | eq(y,x).
-eq(x,y) | -eq(y,z) | eq(x,z).
-eq(u,v) | -P(u,x,y) | P(v,x,y).
-eq(u,v) | -P(x,u,y) | P(x,v,y).
-eq(u,v) | -P(x,y,u) | P(x,y,v).
-eq(u,v) | eq(f(u,x),f(v,x)).
-eq(u,v) | eq(f(x,u),f(x,v)).
-eq(u,v) | eq(g(u),g(v)).
end_of_list.

list(sos).
P(e,x,x).
P(x,e,x).
P(g(x),x,e).
P(x,g(x),e).
P(x,y,f(x,y)).
P(x,x,e).
P(a,b,c).
-P(b,a,c).
end_of_list.
```

A.2 Theorem 2: The Commutator Theorem (P-form)

```
set(hyper_res).
set(back_demod).
set(dynamic_demod_all).
assign(pick_given_ratio,5).
clear(print_kept).
assign(max_mem,20000).
set(control_memory).

list(usable).
-P(x,y,u) | -P(y,z,v) | -P(u,z,w) | P(x,v,w).
-P(x,y,u) | -P(y,z,v) | -P(x,v,w) | P(u,z,w).
-P(x,y,u) | -P(x,y,v) | eq(u,v).
eq(x,x).
-eq(x,y) | eq(y,x).
-eq(x,y) | -eq(y,z) | eq(x,z).
-eq(u,v) | -P(u,x,y) | P(v,x,y).
-eq(u,v) | -P(x,u,y) | P(x,v,y).
-eq(u,v) | -P(x,y,u) | P(x,y,v).
-eq(u,v) | eq(f(u,x),f(v,x)).
```

```
-eq(u,v) | eq(f(x,u),f(x,v)).
-eq(u,v) | eq(g(u),g(v)).
-P(x,x,y) | P(x,y,e).
-P(x,x,y) | P(y,x,e).
end_of_list.

list(sos).
P(e,x,x).
P(x,e,x).
P(g(x),x,e).
P(x,g(x),e).
P(x,y,f(x,y)).
P(a,b,c).
P(c,g(a),d).
P(d,g(b),h).
P(h,b,j).
P(j,g(h),k).
-P(k,g(b),e).
end_of_list.
```

A.3 Theorem 3: $x^2 = x$ Rings Are Commutative (P-form)

```
set(hyper_res).
set(back_demod).
set(dynamic_demod_all).
assign(pick_given_ratio,5).
clear(print_kept).
assign(max_mem,20000).
set(control_memory).

list(usable).
-S(x,y,u) | -S(y,z,v) | -S(u,z,w) | S(x,v,w).
-S(x,y,u) | -S(y,z,v) | -S(x,v,w) | S(u,z,w).
-S(x,y,u) | -S(x,y,v) | eq(u,v).
eq(x,x).
-eq(x,y) | eq(y,x).
-eq(x,y) | -eq(y,z) | eq(x,z).
-eq(u,v) | -S(u,x,y) | S(v,x,y).
-eq(u,v) | -S(x,u,y) | S(x,v,y).
-eq(u,v) | -S(x,y,u) | S(x,y,v).
-eq(u,v) |  eq(j(u,x),j(v,x)).
-eq(u,v) |  eq(j(x,u),j(x,v)).
-eq(u,v) |  eq(g(u),g(v)).
-S(x,y,z) | S(y,x,z).
-P(x,y,u) | -P(y,z,v) | -P(u,z,w) | P(x,v,w).
-P(x,y,u) | -P(y,z,v) | -P(x,v,w) | P(u,z,w).
-P(x,y,v1) | -P(x,z,v2) | -S(y,z,v3) | -P(x,v3,v4) | S(v1,v2,v4).
-P(x,y,v1) | -P(x,z,v2) | -S(y,z,v3) | -S(v1,v2,v4) | P(x,v3,v4).
-P(y,x,v1) | -P(z,x,v2) | -S(y,z,v3) | -P(v3,x,v4) | S(v1,v2,v4).
-P(y,x,v1) | -P(z,x,v2) | -S(y,z,v3) | -S(v1,v2,v4) | P(v3,x,v4).
-P(x,y,u) | -P(x,y,v) | eq(u,v).
-eq(u,v) | -P(u,x,y) | P(v,x,y).
-eq(u,v) | -P(x,u,y) | P(x,v,y).
```

```
-eq(u,v) | -P(x,y,u) | P(x,y,v).
-eq(u,v) | eq(f(u,x),f(v,x)).
-eq(u,v) | eq(f(x,u),f(x,v)).
end_of_list.

list(sos).
S(0,x,x).
S(x,0,x).
S(g(x),x,0).
S(x,g(x),0).
S(x,y,j(x,y)).
P(0,x,0).
P(x,0,0).
P(x,y,f(x,y)).
P(x,x,x).
P(a,b,c).
-P(b,a,c).
end_of_list.
```

A.4 Theorem 4: Equivalential Calculus, XGK → PYO

```
set(hyper_res).
set(back_demod).
set(dynamic_demod_all).
assign(pick_given_ratio,5).
clear(print_kept).
assign(max_mem,20000).
set(control_memory).

list(usable).
-P(x) | -P(e(x,y)) | P(y).
end_of_list.

list(sos).
P(e(x,e(e(y,e(z,x)),e(z,y)))).  % XGK
-P(e(e(e(a,e(b,c)),c),e(b,a))). % the negation of PYO
end_of_list.
```

A.5 Theorem 5: Implicational Calculus Single Axiom, CD-67 (Imp-4)

```
set(hyper_res).
set(back_demod).
set(dynamic_demod_all).
clear(print_kept).
assign(pick_given_ratio,5).
assign(max_mem,20000).
set(control_memory).

list(usable).
-P(x) | -P(i(x,y)) | P(y).
end_of_list.

list(sos).
```

```
P(i(i(i(x,y),z),i(i(z,x),i(u,x)))).
-P(i(i(a,b),i(i(b,c),i(a,c)))).
end_of_list.
```

A.6 Theorem 6: Many-valued Sentential Calculus, CD-57

```
set(hyper_res).
set(back_demod).
set(dynamic_demod_all).
clear(print_kept).
assign(pick_given_ratio,5).
assign(max_mem,20000).
set(control_memory).

list(usable).
-P(x) | -P(i(x,y)) | P(y).
end_of_list.

list(sos).
P(i(x,i(y,x))).
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(i(x,y),y),i(i(y,x),x))).
P(i(i(n(x),n(y)),i(y,x))).
-P(i(i(a,b),i(i(c,a),i(c,b)))).
end_of_list.
```

A.7 Theorem 7: Many-valued Sentential Calculus, CD-60

```
set(hyper_res).
set(back_demod).
set(dynamic_demod_all).
clear(print_kept).
assign(pick_given_ratio,5).
assign(max_mem,20000).
set(control_memory).

list(usable).
-P(x) | -P(i(x,y)) | P(y).
end_of_list.

list(sos).
P(i(x,i(y,x))).
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(i(x,y),y),i(i(y,x),x))).
P(i(i(n(x),n(y)),i(y,x))).
-P(i(i(a,b),i(n(b),n(a)))).
end_of_list.
```

B Inputs to OTTER 2.2 for the Equality Set

B.1 Theorem EQ-1: The Commutator Theorem

```
set(knuth_bendix).
set(index_for_back_demod).
set(process_input).
assign(max_mem, 16000).
set(control_memory).
set(lex_rpo).

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

lex([a,b,e,f(x,x),g(x),h(x,x)]).
lrpo_lr_status([f(x,x)]).

list(usable).
(x = x).
(f(e, x) = x).
(f(g(x), x) = e).
(f(f(x, y), z) = f(x, f(y, z))).
(h(x, y) = f(x, f(y, f(g(x), g(y))))).
end_of_list.

list(sos).
(f(x, f(x, x)) = e).
(h(h(a, b), b) != e).
end_of_list.
```

B.2 Theorem EQ-2: Robbins Algebra, $(\exists c, c + c = c) \rightarrow$ Boolean

```
set(knuth_bendix).
set(index_for_back_demod).
set(process_input).
assign(max_mem, 16000).
set(control_memory).
set(lex_rpo).

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

lex([a,b,c,o(x,x),n(x)]).
lrpo_lr_status([o(x,x)]).

list(usable).
(x = x).
(o(x,y) = o(y,x)).
(o(o(x,y),z) = o(x,o(y,z))).
(n(o(n(o(x,y)),n(o(x,n(y))))) = x).
end_of_list.
```

```
list(sos).
(o(c,c) = c).
(o(n(o(a,n(b))),n(o(n(a),n(b)))) != b).   % denial of Huntington axiom

end_of_list.
```

B.3 Theorem EQ-3: On Ternary Boolean Algebra

```
set(knuth_bendix).
set(index_for_back_demod).
set(process_input).
assign(max_mem, 16000).
set(control_memory).
set(lex_rpo).

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

lex([a,b,c,f(x,x,x),g(x)]).
lrpo_lr_status([f(x,x,x)]).

list(usable).
(x = x).
end_of_list.

list(sos).
(f(f(v,w,x),y,f(v,w,z)) = f(v,w,f(x,y,z))).
(f(y,x,x) = x).
(f(x,x,y) = x).
(f(g(y),y,x) = x).
(f(a,g(a),b) != b).
end_of_list.
```

B.4 Theorem EQ-4: Group Theory Single Axiom

```
set(knuth_bendix).
set(index_for_back_demod).
set(process_input).
assign(max_mem, 16000).
set(control_memory).
set(lex_rpo).

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

lex([a,b,c,f(x,x),i(x)]).
lrpo_lr_status([f(x,x)]).

list(usable).
(x = x).
```

```
end_of_list.

list(sos).
(f(x,i(f(f(i(f(i(y),f(i(x),w))),z),i(f(y,z))))) = w).
(f(a,f(b,c)) != f(f(a,b),c)).
end_of_list.
```

B.5 Theorem EQ-5: On Wajsberg Algebra

```
set(knuth_bendix).
set(index_for_back_demod).
set(process_input).
assign(max_mem, 16000).
set(control_memory).
set(lex_rpo).

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

lex([a,b,T,i(x,x),n(x)]).
lrpo_lr_status([i(x,x)]).

list(usable).
(x = x).
end_of_list.

list(sos).
(i(T,x) = x).
(i(i(x,y),i(i(y,z),i(x,z))) = T).
(i(i(x,y),y) = i(i(y,x),x)).
(i(i(n(x),n(y)),i(y,x)) = T).

(i(i(i(a,b),i(b,a)),i(b,a)) != T).
end_of_list.
```

References

[1] E. Lusk and W. McCune. Experiments with ROO, a parallel automated deduction system. In B. Fronhöfer and G. Wrightson, editors, *Parallelization in Inference Systems, Lecture Notes in Artificial Intelligence, Vol. 590*, pages 139–162, New York, 1992. Springer-Verlag.

[2] E. Lusk, W. McCune, and J. Slaney. ROO—a parallel theorem prover. Tech. Memo ANL/MCS-TM-149, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1991.

[3] E. Lusk and R. Overbeek. The automated reasoning system ITP. Tech. Report ANL-84/27, Argonne National Laboratory, Argonne, Ill., April 1984.

[4] J. McCharen, R. Overbeek, and L. Wos. Complexity and related enhancements for automated theorem-proving programs. *Computers and Mathematics with Applications*, 2:1–16, 1976.

[5] J. McCharen, R. Overbeek, and L. Wos. Problems and experiments for and with automated theorem-proving programs. *IEEE Transactions on Computers*, C-25(8):773–782, August 1976.

[6] W. McCune. OTTER 2.0 Users Guide. Tech. Report ANL-90/9, Argonne National Laboratory, Argonne, Ill., March 1990.

[7] W. McCune. What's New in OTTER 2.2. Tech. Memo ANL/MCS-TM-153, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., July 1991.

[8] W. McCune and L. Wos. The absence and the presence of fixed point combinators. *Theoretical Computer Science*, 87:221–228, 1991.

[9] W. McCune and L. Wos. Experiments in automated deduction with condensed detachment. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction, Lecture Notes in Artificial Intelligence, Vol. 607*, pages 209–223, New York, June 1992. Springer-Verlag.

[10] B. Smith. Reference manual for the environmental theorem prover: An incarnation of AURA. Tech. Report ANL-88-2, Argonne National Laboratory, Argonne, Ill., March 1988.

[11] Hantao Zhang. Automatic proofs of equality problems in Overbeek's competition. *Journal of Automated Reasoning*, this issue.