Quasigroups and Related Systems **10** (2003), 95 – 114

See Otter digging for algebraic pearls

J. D. Phillips

Abstract

We give an introduction to Otter for the practicing loop theorist. Although this paper is self-contained, it is intended as a supplement to the author's Otter workshop at Loops'03, Prague, August 2003.

1. Introduction

Otter is an automated reasoning tool developed by William McCune (Argonne) which has proven to be effective at equational reasoning [17]. McCune, R. Padmanabhan, Larry Wos, and many others have made good use of Otter in many areas of algebra and logic, most notably in lattice theory [17], [19], [20], [23]. For instance, in 1997, McCune used Otter to help him solve one of the most celebrated open problems in mathematics, the so-called Robbins Conjecture [18]. McCune's solution generated a buzz loud enough to warrant press in the New York Times, a rare event indeed in mathematics! Until recently, though, there were comparatively few Otter applications in quasigroup and loop theory. Ken Kunen was the pioneer in this regard. He used Otter to help show that Moufang quasigroups are loops [14], among a number of other important early papers [15], [16]. Recently, various subsets of {Kepka, Kinyon, Kunen, Phillips} have used Otter to help

- solve an old open problem in loop theory, originating in Bruck's early work, by showing that diassociative A-loops are Moufang [7],
- solve a long-standing open problem in quasigroup theory, the first open problem listed in Belousov's book [2], by showing that F-quasigroups are isotopic to Moufang loops [6], [12],

2000 Mathematics Subject Classification: 20N05
Keywords: otter, automated theorem proved, loop, quasigroup

- show that commutants of Bol loops of odd order are subloops [11],
- solve other problems in loop theory [8], [9].

Thus, it is clear that Otter is a powerful tool for quasigroup and loop theorists. It is also an unfertilized tool. And so the purpose of this paper is to give the working quasigroup and loop theorist enough of a background to teach himself Otter, if he is so inclined.

A word of caution. This paper is meant only as a primitive introduction. It is not intended as an Otter manual. Herein, we limit ourselves to a few brief remarks on five basic Otter input files, two complete Otter proofs, and a few "translated" proofs. As with most software, the user's ultimate success and sense of agency depends mostly on his or her sense of adventure and willingness to play around with and explore the program for himself or herself. This especially obtains with Otter, which has a steep learning curve, and (with all due respect to McCune) a hard-to-read manual. All this is to say that my disciplinary expertise lies far afield from automated theorem provers. This paper, then, is meant to be nothing more than a loop theorist's rudimentary introduction to Otter, an invitation to other loop theorists to explore Otter for themselves.

2. Quasigroups and loops

Otter's prowess with equational reasoning means that inquiries involving quasigroups and loops are particularly well suited to this tool because they can be defined equationally. To wit, a *quasigroup* is a set Q together with three binary operations $\cdot$, $\backslash$, and $/$ satisfying: $x\backslash(x\cdot y) = (x\cdot y)/y = x$ and $x\cdot(x\backslash y) = (x/y)\cdot y = x$. See [2], [3], [4], and [21] for solid introductions to the theory of quasigroups (and loops). We often denote $\cdot$ by juxtaposition.

In a quasigroup Q, define left and right translations in the customary manner: $xR(y) = yL(x) = xy$. The *multiplication group* of Q, denoted $\mathrm{Mlt}(Q)$, is the group of permutations on Q generated by all of the right and left translations. Clearly $\mathrm{Mlt}(Q)$ acts as a permutation group on Q.

A *loop* is a quasigroup with a 2-sided neutral element, 1: $x\cdot 1 = 1\cdot x = x$. The *commutant* of a loop, L, is the subset $\mathrm{C}(L) = \{a : ax = xa, \forall x\}$. It is not necessarily a subloop [11]. We note that the commutant is known by other names in the literature, for instance, *centrum* and *Moufang center*. The *left nucleus* of a loop L is the subloop $\mathrm{N}_\lambda(L) = \{a : a\cdot xy = ax\cdot y, \forall x, y\}$. The middle and right nuclei, $\mathrm{N}_\mu(L)$ and $\mathrm{N}_\rho(L)$ are defined analogously. The

nucleus, $\mathrm{N}(L)$, of L is the subloop given by $\mathrm{N}(L) = \mathrm{N}_\lambda(L) \cap \mathrm{N}_\mu(L) \cap \mathrm{N}_\rho(L)$. The *center*, $\mathrm{Z}(L)$, of a loop L is the subloop given by $\mathrm{Z}(L) = \mathrm{N}(L) \cap \mathrm{C}(L)$.

If L is a loop, the stabilizer in $\mathrm{Mlt}(L)$ of 1 is called the *inner mapping group* of L, and is denoted by $\mathrm{I}(L)$. A subloop is normal if it is invariant as a set under each inner mapping. The center, $\mathrm{Z}(L)$, is a normal subloop of L. For many important varieties of loops, e.g., Moufang loops, the nucleus, $\mathrm{N}(L)$, is normal. A loop is called an *A-loop* if every inner mapping is an automorphism. Thus, A-loops are generalizations of both groups and commutative Moufang loops. They are also equationally-defined, since $\mathrm{I}(L)$ is generated by $R(x,y) = R(x)R(y)R(xy)^{-1}, L(x,y) = L(x)L(y)L(xy)^{-1}$, and $T(x) = R(x)L(x)^{-1}$ [3].

A loop is *diassociative* if any two elements generate a group; it is not known if the variety of diassociative loops is finitely based. A *Moufang loop* is a loop satisfying the identity $(xy \cdot x)z = x(y \cdot xz)$; there are many well known equivalent identities. Define the right and left inverse maps, ρ and λ on a loop L by $x^\rho = x\backslash 1$ and $x^\lambda = 1/x$. A loop has the *right inverse property* if it satisfies $y/x = yx^\lambda$. There is an obvious left version of this identity. If a loop satisfies both the left and right inverse properties, it has the *inverse property*. If the right and left inverse maps coincide we usually denote this map by x^{-1}. A loop has the *anti-automorphic inverse property* if it satisfies $(xy)^{-1} = y^{-1}x^{-1}$. The *automorphic inverse property* is given by $(xy)^{-1} = x^{-1}y^{-1}$. A loop has the *weak inverse property* if it satisfies $x \cdot (yx)^{-1} = y^{-1}$; it is flexible if it satisfies $x \cdot yx = xy \cdot x$. A loop is a (left) *Bol loop* if it satisfies the following identity $x(y \cdot xz) = (x \cdot yx)z$. A (left) *Bruck loop* is a (left) Bol loop with the automorphic inverse property. A *conjugacy closed loop* is a loop in which the right translations are closed under conjugation and the left translations are closed under conjugation [9]. An *Osborn loop* satisfies (equivalent) generalized Moufang identities: $(x[yx^\lambda \cdot x]) \cdot (zx) = x(yz \cdot x)$ and $(xy) \cdot ([x \cdot x^\rho z]x) = (x \cdot yz)x$. These include Moufang loops and conjugacy closed loops as special cases [1]. A *principal isotope* of a loop is another operation on the same underlying set of the form $x \cdot_{a,b} y := (x/a) * (b\backslash y)$.

3. Otter

Otter is an automated theorem prover. It can be downloaded from the site `http://www-unix.mcs.anl.gov/AR/otter/`. There are UNIX, PC, and Mac versions; all are easy to install. All versions come bundled with MACE, a finite model building program. This paper is about Otter; we will leave

MACE and finite model builders for another time. Otter implements the Knuth-Bendix algorithm. More about this, and indeed about most of Otter's technical specifications can found in [20]. There is a large, and growing, literature on Otter, including a number of books and manuals. Pointers to all of them can found in [20].

I usually use Otter on a UNIX platform. My UNIX machine is a relatively fast, small-sized server—a 1 gigahertz Pentium 3 machine with 765 megs of RAM. So keep in mind that some of the Otter statistics I give below may vary a bit if you use a different platform, or if your UNIX machine is faster or slower than mine. There are many styles for Otter input files. For a sample of some that are different from mine see [20].

Otter produces proofs that are often complicated, opaque, and very nearly intractable to humans. A big part, then, of our Otter assisted inquiries is translating the proofs that Otter generates into a form that a human can understand. We'll have more to say about this below. Here we note that the practice of translating Otter proofs is not universal. There are many examples of published Otter proofs with no accompanying translation [20]. One of the reasons that we translate our Otter proofs is that most of our results are only Otter assisted. By that, I mean that we find general patterns by examining a sequence of translated Otter proofs, and then construct general proofs sans Otter. This is especially true of proofs of theorems that make claims about all integers, since Otter cannot produce such proofs. Of course, translating Otter proofs is decidedly labor intensive. But the pay-off is greater insight—untranslated Otter proofs reveal only *that* a theorem is true, not *why* it is true.

4. An Input file

In this section we examine a typical Otter input file. Most of my input files are structured like the one in this section. This is mainly because this format has proven effective for me; I don't know if it's the "best".

First note what the file below is asking Otter to prove: If L is a Moufang loop, and if C is a commutant element, then $(C, x^3, y) = 1$. I generally place this "target" in the penultimate line of the input file. Now let's examine the file's features.

The first line tells Otter that $*$ is a binary operation. The number 400 in this line is a code that tells Otter not to suppress parantheses, which is obviously crucial in proofs about loops. The next five lines are technical – Otter is to use the Knuth-Bendix algorithm, etc. I rarely change these lines

in my Otter files.

The next line declares $1, A, B$, and C as constants, $a(\_,\_,\_)$ as a ternary function (the associator, as we shall see), $i(\_)$ as a unary function (the inverse), and $*$ (again) as a binary operation. This line also specifies the lexicographical order. Otter tries to express functions and operations which are lower in the lexicographical order in terms of those which are higher, so it is particularly sensitive to the order specified here.

The next nine lines tell Otter how to limit the length of the identities it deduces, etc. And as with the five lines above, I rarely change any of them, except the "max_weight" specification.

In the next line "sos" stands for "set of support". The first six lines following "list(sos)" tell Otter that the binary operation $*$ gives an inverse property loop. [As with TEX, lines beginning with % are ignored by Otter; we use them for documentation.] The next line tells Otter that the loop satisfies the anti-automorphic inverse property. Of course, this property follows from the inverse property. But it is sometimes useful to add clauses which follow from the others in order to help Otter in its search. The addition of the anti-automorphic inverse property in this input file is a typical example.

The next three lines are axiomatizations of three versions of the Moufang law, and the following line stipulates that the constant "C", which we officially declared in the lexicographical ordering, is in the commutant. The next line defines the associator function, and the final line tells Otter to stop making deductions when it produces the desired identity.

Finally we note that lower case letters at the end of the alphabet are understood by Otter to be free variables. Let's have a look at the input file:

```
op(400, xfx, *).

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

lex([1,A,B,C,a(_,_,_),_*_,i(_)]).

assign(max_weight, 60).
assign(pick_given_ratio, 3).
assign(change_limit_after, 100).
assign(new_max_weight, 21).
assign(neg_weight, -4).
```

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

list(sos).

% Inverse property loop
1 * x = x.
x * 1 = x.
i(x) * x = 1.
x * i(x) = 1.
i(x) * (x * y) = y.
(y * x) * i(x) = y.

% Anti-automorphic inverse property
i(x * y) = i(y) * i(x).

% Moufang
((x * y) * x) * z = x * (y * (x * z)).
x * ((y * z) * x) = (x * y) * (z * x).
((z * x) * y) * x = z * (x * (y * x)).

% C in C(L)
x * C = C * x.

% Associator a(x,y,z) defined
a(x,y,z) = i(x * (y * z)) * ((x * y) * z).

% Theorem?
a(C,A * (A * A),B) != 1.

end_of_list.
```

It took Otter 1 hour, 13 minutes, and 34 seconds on my UNIX machine to find a proof. This Otter proof is 15 printed pages long. In the language of Otter the "Length of proof" is 276. The "Level of proof" is 33 (cf: the much quicker and shorter proofs in [20]). The upshot is that this proof is long enough to be a burden to translate into the form of a more traditional proof that is palatable to human intelligence. So it's natural to ask if we can coax Otter to produce a shorter proof. As we shall see in the next section, indeed we can.

5. A better Input file, proof, and translation

Here is a slightly different input file that asks Otter to prove the same thing as the input file in the previous section, but without appealing to a specially designated associator function:

```
op(400, xfx, *).

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

lex([1,A,B,C,_*_,i(_)]).

assign(max_weight, 60).
assign(pick_given_ratio, 3).
assign(change_limit_after, 100).
assign(new_max_weight, 21).
assign(neg_weight, -4).

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

list(sos).

% Inverse property loop
1 * x = x.
x * 1 = x.
i(x) * x = 1.
x * i(x) = 1.
i(x) * (x * y) = y.
(y * x) * i(x) = y.

% Anti-automorphic inverse property
i(x * y) = i(y) * i(x).

% Moufang
((x * y) * x) * z = x * (y * (x * z)).
x * ((y * z) * x) = (x * y) * (z * x).
((z * x) * y) * x = z * (x * (y * x)).

% C in C(L)
```

```
x * C = C * x.

% Theorem?
C * (((A * A) * A) * B) != (C * ((A * A) * A)) * B.

end_of_list.
```

The Otter proof for this input file is significantly shorter than is the Otter proof of the very similar input file in the last section. In the technical language of Otter, "Length of proof" is 28; "Level of proof" is 7. The point is that Otter is decidedly sensitive to small perturbations in the input file (lexicographical order, extra functions, max_weight, etc.) Here is the proof, followed by the standard compilation of Otter statistics; note in particular how quickly Otter found a proof (7 seconds):

```
---------------- PROOF ----------------
3,2 [] 1*x=x.
5,4 [] x*1=x.
6 [] i(x)*x=1.
10 [] i(x)* (x*y)=y.
12 [] (x*y)*i(y)=x.
17,16 [] ((x*y)*x)*z=x* (y*(x*z)).
19 [] ((x*y)*z)*y=x* (y*(z*y)).
21 [] x*C=C*x.
22 [] C* (((A*A)*A)*B)!=(C*((A*A)*A))*B.
23 [copy,22,demod,17,flip.1] (C*((A*A)*A))*B!=C* (A*(A*(A*B))).
25 [copy,21,flip.1] C*x=x*C.
28 [para_into,10.1.1.2,25.1.1] i(C)*(x*C)=x.
35,34 [para_into,10.1.1.2,6.1.1,demod,5] i(i(x))=x.
38 [para_into,12.1.1.1,21.1.1] (C*x)*i(C)=x.
96 [para_into,16.1.1.1.1,10.1.1] (x*i(y))*z=i(y)*((y*x)*(i(y)*z)).
98,97 [para_into,16.1.1.1.1,4.1.1,demod,3] (x*x)*y=x* (x*y).
106,105 [para_into,16.1.1,4.1.1,demod,5] (x*y)*x=x*(y*x).
111 [copy,96,flip.1] i(x)*((x*y)*(i(x)*z))= (y*i(x))*z.
112 [back_demod,23,demod,98] (C*(A*(A*A)))*B!=C*(A*(A*(A*B))).
117,116 [back_demod,16,demod,106] (x*(y*x))*z=x*(y*(x*z)).
205 [para_into,19.1.1.1.1,21.1.1] ((C*x)*y)*C=x*(C*(y*C)).
221 [para_into,19.1.1.1,38.1.1,flip.1] C*(x*(i(C)*x))=x*x.
232 [para_into,19.1.1.1,4.1.1,demod,3] (x*y)*y=x*(y*y).
262 [para_from,19.1.1,28.1.1.2] i(C)*(x*(C*(y*C)))=(x*C)*y.
269 [copy,262,flip.1] (x*C)*y=i(C)*(x*(C*(y*C))).
448 [para_into,97.1.1,21.1.1] C*(x*x)=x*(x*C).
453 [copy,448,flip.1] x*(x*C)=C*(x*x).
551 [para_into,232.1.1.1,232.1.1] (x*(y*y))*y=(x*y)*(y*y).
```

```
 592,591 [para_into,232.1.1,19.1.1,flip.1] (x*y)*(y*y)=x*(y*(y*y)).
 596,595 [back_demod,551,demod,592] (x*(y*y))*y=x*(y*(y*y)).
 891 [para_from,453.1.1,10.1.1.2] i(x)*(C*(x*x))=x*C.
 3225,3224 [para_into,111.1.1.2.1,891.1.1,demod,35,35,35,596,flip.1]
     (C*(x*(x*x)))*y=x*((x*C)*(x*y)).
 3297 [back_demod,112,demod,3225] A*((A*C)*(A*B))!=C*(A*(A*(A*B))).
 5663 [para_into,205.1.1,21.1.1] C*((C*x)*y)=x*(C*(y*C)).
 6858 [para_into,5663.1.1.2.1,221.1.1,demod,98,117]
      C*(x*(x*y))=x*(i(C)*(x*(C*(y*C)))).
 6865 [copy,6858,flip.1] x*(i(C)*(x*(C*(y*C))))=C*(x*(x*y)).
 7052 [para_from,269.1.1,3297.1.1.2]
      A*(i(C)*(A*(C*((A*B)*C))))!=C*(A*(A*(A*B))).
 7053 [binary,7052.1,6865.1] $F.
 ------------ end of proof -------------

 Search stopped by max_proofs option.

 ============ end of search ============

-------------- statistics -------------
clauses given                277
clauses generated          66537
  para_from generated      30209
  para_into generated      36328
demod & eval rewrites     262258
clauses wt,lit,sk delete   31763
tautologies deleted            0
clauses forward subsumed   33438
  (subsumed by sos)         4720
unit deletions                 0
factor simplifications         0
clauses kept                4519
new demodulators            2532
empty clauses                  1
clauses back demodulated    1189
clauses back subsumed        220
usable size                  244
sos size                    2866
demodulators size           1657
passive size                   0
hot size                       0
Kbytes malloced             5779

----------- times (seconds) -----------
user CPU time          7.51          (0 hr, 0 min, 7 sec) system
```

```
CPU time        0.41          (0 hr, 0 min, 0 sec) wall-clock time
16              (0 hr, 0 min, 16 sec)
 .
 .
 .
That finishes the proof of the theorem.
```

This proof is clearly much easier to translate into a traditional proof than is the fifteen page proof for the input file from the previous section. In the proof, "para_into" and "para_from" are short for "paramodulation into" and "paramodulation from", and they are the key steps in any Otter proof. Very crudely, paramodulation is an inference rule that combines variable instantiation and equality substitution into one step [20]. Here is the translated proof, buffed and polished:

If c is in $\mathrm{C}(L)$ then $x^2 \cdot cy = cx^2c^{-1} \cdot yc = c(x^2c^{-1} \cdot y)c = c(xc^{-1}x \cdot y)c = c(x[c^{-1} \cdot xy])c = (cx)([c^{-1} \cdot xy] \cdot c) = xc \cdot xy$. Now multiply both sides on the left by x.

6. Translation intemperance

Otter proofs can be vexing in other ways. Consider the following theorem:

Theorem 6.1. *If L is a Moufang loop, and if c is in the commutant of L, then the set $\{b : (c, b, x) = 1, \forall x \in L\}$ is a subloop.*

Here is an input file asking Otter to prove this theorem:

```
op(400, xfx, *).

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

lex([1,A,B,C,D,_*_,i(_)]).

assign(max_weight, 60).
assign(pick_given_ratio, 3).
assign(change_limit_after, 100).
assign(new_max_weight, 21).
assign(neg_weight, -4).
```

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

list(sos).

% Inverse property loop
1 * x = x.
x * 1 = x.
i(x) * x = 1.
x * i(x) = 1.
i(x) * (x * y) = y.
(y * x) * i(x) = y.

% Anti-automorphic inverse property
i(x * y) = i(y) * i(x).

% Moufang
((x * y) * x) * z = x * (y * (x * z)).
x * ((y * z) * x) = (x * y) * (z * x).
((z * x) * y) * x = z * (x * (y * x)).

% C in C(L)
x * C = C * x.
(C * C) * (x * y) = (C * x) * (C * y).

% assumption
(C * A) * x = C * (A * x).
(C * B) * x = C * (B * x).

% Theorem?
(C * (A * B)) * D != C * ((A * B) * D).
end_of_list.
```

Otter finds a proof in 4 minutes, 43 seconds. Length of proof is 170; level of proof is 23. The Otter proof, though, is not particularly enlightening, so I have not included it here. It is tedious and irksome to translate. Here is the translation, which itself is not very enlightening:

If $(c,b,x) = 1 = (c,f,x)$, then
$c(fb\cdot x) = c(x^{-1}[xf\cdot bx]) = c^{-2}(c^3[x^{-1}(xf\cdot bx)]) = c^{-2}(x^{-1}[c^3(xf\cdot bx)]) = c^{-2}(x^{-1}[c(cxf\cdot bxc)]) = c^{-2}(x^{-1}[c^{-1}(c[(cxf\cdot bxc)c])]) =$
$c^{-1}[(c^{-1}x^{-1}c^{-1})(c[cxf\cdot bxc]c)] = [(c^{-1}x^{-1}c^{-1})([cxc\cdot f]\cdot[b\cdot cxc])]c^{-1} = (fb\cdot cxc)c^{-1} = ([(fb\cdot c)x]c)c^{-1} = (c\cdot fb)x.$ That is, $(c,fb,x) = 1$.

But here is a better proof, a basic proof that does not rely on Otter at all:

Note that $(x, y, z) = 1$ is invariant under permutations of x, y, z. Now note that $(a, x, y) = 1$ if and only if $aR(x, y) = a$. But each $R(x, y)$ is a pseudo-automorphism, and the set of fixed points of any family of pseudo-automorphisms is a subloop.

The point of all of this is to offer a simple word of warning. The power of Otter is seductive. As your appreciation of its virtues deepens, it's almost inevitable that you will come increasingly to rely on it, so much so that it's likely you will have Otter prove things—and that you will then devote a considerable amount of time to translating the resulting complicated Otter proof—that are quite easily proved without using Otter at all, as above.

7. Translation transcendence

In 1978, Stephen Doro asked about conditions under which a Moufang loop's commutant is normal [5]. It seemed that not much work had been done in the intervening years toward answering Doro. So we began exploring possible responses to his question. Eventually, we found a very long and complicated Otter proof of the following:

Theorem 7.1. *If c is a commutant element in a Moufang loop, then $cR(a, b)^3 = c$.*

Here is an early input file asking Otter to prove this:

```
op(400, xfx, *).

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

lex([1,A,B,C,D,E,F,_*_,i(_)]).

assign(max_weight, 50).
assign(pick_given_ratio, 3).
assign(change_limit_after, 100).
assign(new_max_weight, 21).
assign(neg_weight, -4).

list(usable).
```

```
(x = x).
end_of_list.

list(sos).

% Inverse property loop
1 * x = x.
x * 1 = x.
i(x) * x = 1.
x * i(x) = 1.
i(i(x)) = x.
i(x) * (x * y) = y.
(y * x) * i(x) = y.

% Anti-automorphic inverse property
i(x * y) = i(y) * i(x).

% Moufang
((x * y) * x) * z = x * (y * (x * z)).
(x * y) * (z * x) = x * ((y * z) * x).
((x * y) * z) * y = x * (y * (z * y)).

% C in C(L)
C * x = x * C.

% R(A,B) and its powers act on C
((C * A) * B = D * (A * B).
((D * A) * B = E * (A * B).
((E * A) * B = F * (A * B).

% Theorem?
F != E.
end_of_list.
```

This seemed like an exciting result. But the 50 page Otter proof was nearly intractable, so we worked hard to shorten it. Eventually we whittled it down to something that, only with great effort, we could translate. Here is our translated proof, which is itself dense and nearly opaque:
$cR(a,b)R(a)R(b) = ([(ca \cdot b)(ab)^{-1}]a)b = (cb \cdot a)(a^{-1}b^{-1}ab) =$
$a(a^{-1}[(cb{\cdot}a)(a^{-1}b^{-1}ab)]) = a([a^{-1}{\cdot}cb][b^{-1}ab]) = a([b(b^{-1}a^{-1}{\cdot}c)b][b^{-1}ab]) =$
$a([b(c \cdot b^{-1}a^{-1})b][b^{-1}ab]) = a([bc \cdot b^{-1}a^{-1}b][b^{-1}ab]) = a \cdot cb =$
$(ab^{-1}a^{-1})[a^{-1}(a[(aba^{-1})(a \cdot cb)])] = (ab^{-1}a^{-1}) \cdot (a^{-1}(a^2b \cdot cb)) =$
$(ab^{-1}a^{-1})dot[a^{-1}([(a^2b \cdot cb)b]b^{-1})] = (ab^{-1}a^{-1})[a^{-1}([a^2 \cdot cb^3]b^{-1})] =$
$(ab^{-1}a^{-1})[a^{-1}([a^2c \cdot b^3]b^{-1})] = (ab^{-1}a^{-1})[a^{-1}(a^2c \cdot b^2)] =$

$(ab^{-1}a^{-1})[a^{-1}(a[c\cdot ab^2])] = (ab^{-1}a^{-1})(c\cdot ab^2) =$
$([(a[b^{-1}a^{-1}])(c\cdot ab^2)]\cdot[b^{-1}a^{-1}])(ab) = (a[b\cdot(c\cdot b^{-1}a^{-1})])(ab) =$
$[(a[b(c\cdot b^{-1}a^{-1})]a)a^{-1}](ab) = [([ab][(c\cdot b^{-1}a^{-1})a])a^{-1}](ab) =$
$([([([ab][(c\cdot b^{-1}a^{-1})a])b]b^{-1})]a^{-1})(ab) = ([(a[(b\cdot[b^{-1}a^{-1}\cdot c])(ab)])b^{-1}]a^{-1})(ab)$
$= [([a([ab]([b^{-1}a^{-1}][b([b^{-1}a^{-1}]\cdot c)])[ab])])b^{-1})a^{-1}]\cdot(ab) =$
$([(a[(ab)(b^{-1}a^{-2}\cdot c)(ab)])b^{-1}]a^{-1})(ab) = ([(a[a^{-1}(c\cdot ab)])b^{-1}]a^{-1})(ab) =$
$[([(c\cdot ab)b^{-1}]a^{-1})(ab) = [([bcb^{-1}\cdot ab]b^{-1})a^{-1}](ab) = [([b(cb^{-1}\cdot a)b]b^{-1})a^{-1}](ab)$
$= ([b(cb^{-1}\cdot a)]a^{-1})(ab) = [(cb\cdot b^{-1}a)a^{-1}](ab) = [([(cb\cdot a)(a^{-1}b^{-1})]a)a^{-1}](ab)$
$= [(cb\cdot a)(a^{-1}b^{-1})](ab) = cR(b,a)R(ab).$

And since $R(a,b)^{-1} = cR(b,a)$ [3] we have shown that $cR(a,b)^2 = cR(a,b)^{-1}$, that is, $cR(a,b)^3 = c$.

But finally, after all of this hard work we rediscovered some old results buried in [3] that render the proof of this theorem trivial:

First note that in any Moufang loop, $R(x,y) = L(x^{-1},y^{-1})$. So here we work with left inner mappings. The associator (c,x,y) lies in the center of the subloop $< c,x,y >$. So for all $n, cL(x,y)^n = [c\cdot(c,y,x)^{-1}]L(x,y)^{n-1} = cL(x,y)^{n-1}\cdot(c,y,x)^{-1}$. Thus $cL(x,y)^3 = c\cdot(c,y,x)^{-3} = c$.

The point of all of this is again clear. As above, Otter is seductive, and at times even bewitching. But still, a curse can be salutary: reproving known, even classic, results with Otter, and then carefully translating the proofs, as above, can sometimes lead to useful insights. In the case of our long translation above, we stumbled upon the following charming identity, that appears to have gone unnoticed until now:

Lemma 7.2. $(ca\cdot b^{-1})(ba^{-1}) = (cb\cdot a)(a^{-1}b^{-1})$.

The proof of this lemma, and indeed much more, can found in [13].

8. A quick tease

In this section we underscore the efficacy of Otter in inquiries involving quasigroups and loops by mentioning a few not-yet-published results, to whet the reader's appetite.

***A*-loops.** We significantly improved our main result in [7] by proving the following theorem which we have not yet published.

Theorem 8.1. *An inverse property loop with*
(*a*) *each* $T(x) = R(x)L(x)^{-1}$ *an automorphism, and*
(*b*) *satisfying* $[R(xy),R(x)R(y)] = I$

is Moufang.

From this, our inquiries led us to the following conjecture.

Conjecture 8.2. *The following varieties are distinct:*
1. *Moufang loops satisfying* (a)
2. *Moufang loops satisfying* (a) *and* (b)
3. *Moufang A-loops.*

Finally, regarding A-loops, here is a sampling of our not-yet-published results on general A-loops:

1. Every A-loop with the weak inverse property is Moufang.
2. Every A-loop has the anti-automorphic inverse property.
3. An A-loop Q with centrally endomorphic cubing and with $Q/C(Q)$ commutative is Moufang. And so clearly commutative A-loops of exponent 3 are Moufang.
4. Every finite simple A-loop of odd order is of exponent p for some odd prime p.

Bruck loops. Turning our attention elsewhere, we note that Bruck loops are a generalization of commutative Moufang loops. In fact, it is well known that Bruck loops of exponent 3 are actually commutative Moufang loops. It turns out that a bit more is true. We have shown that in a Bruck loop of exponent 6 elements of order 3 are in the commutant. Elsewhere, the next theorem gives a new Otter-assisted result that we haven't yet published and that we think reveals something deep about the structure of (finite) Bruck loops:

Theorem 8.3. *In a Bruck loop, elements of order a power of* 2 *commute with elements of odd order.*

Osborn loops. Commutative Moufang loops are perhaps the most widely studied variety of nonassociative loops because of their connections with combinatorics and geometry (e.g., Steiner triple systems). One of the most striking algebraic features of commutative Moufang loops is that they have centrally endomorphic cubing. This fact underscores the allure of the following generalization, which we found with the help of Otter and which have not yet published.

Theorem 8.4. *An Osborn loop with the automorphic inverse property has centrally endomorphic cubing.*

There are indeed examples of nonMoufang Osborn loops with the automorphic inverse property. We think there is great potential for fruitful inquiry into the structure of Osborn loops.

Isotopic Invariance. An important program in loop theory is to find *isotopic invariant* properties, that is, properties which hold in all isotopes of a loop. For example, the property of being a Moufang loop is isotopically invariant. In a series of papers culminating in [22], Syrbu showed that inverse property loops with isotopically invariant flexibility share many properties with Moufang loops. Recently, Otter helped us prove that this is not surprising [10]:

Theorem 8.5. *Inverse property loops with isotopically invariant flexibility are Moufang.*

We also find some new varieties of diassociative loops in [10], complementing the main result in [9].

Other varieties. Finally, in this section, we mention that we have scores of not-yet-published Otter assisted results about i-loops, rectangular loops, trimedial quasigroups, to name just a few, as well as some deep results about nilpotency in many varieties of loop.

9. Coda, a quasigroup input file

The four input files we have examined up to this point all have asked Otter to prove theorems about Moufang loops. And as we have seen, it is convenient to axiomatize Moufang loops as sets with a single binary operation satisfying certain identities (together with a unary operation giving inverses). But for general quasigroups, it is necessary to use three binary operations to axiomatize the variety. So here we give a typical Otter quasigroup input file. The file below asks Otter to prove that quasigroups which are both left and middle semimedial must also be right semimedial. This, by the way, is a new result [6].

```
op(400, xfx, *).
op(400, xfx, \).
op(400, xfx, /).

set(knuth_bendix).
clear(print_new_demod).
clear(print_back_demod).
```

```
clear(print_back_sub).
clear(print_kept).

lex([A,B,C,_*_,_\_,_/_]).

assign(max_weight, 50).
assign(pick_given_ratio, 3).
assign(change_limit_after, 100).
assign(new_max_weight, 21).
assign(neg_weight, -4).

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

list(sos).

% Quasigroup
x * (x \ y) = y.
x \ (x * y) = y.
(x * y) / y = x.
(x / y) * y = x.

% Middle semimedial
(x * y) * (z * x) = (x * z) * (y * x).

% Left semimedial
(x * x) * (y * z) = (x * y) * (x * z).

% Right semimedial?
(B * C) * (A * A) != (B * A) * (C * A).
end_of_list.
```

Otter finds the proof very quickly—in 30 seconds. Here is the short proof, sans statistics, which you might enjoy translating for yourself.

```
Length of proof is 28.  Level of proof is 8.

---------------- PROOF ----------------
 3,2 [] x* (x\y)=y.
 4 [] x\ (x*y)=y.
 6 [] (x*y)/y=x.
 9,8 [] (x/y)*y=x.
 10 [] (x*y)* (z*x)= (x*z)* (y*x).
 11 [] (x*x)* (y*z)= (x*y)* (x*z).
```

```
 12 [] (B*C)* (A*A)!= (B*A)* (C*A).
 17,16 [para_from,8.1.1,4.1.1.2] (x/y)\x=y.
 21 [para_into,10.1.1.1,2.1.1] x* (y*z)= (z*y)* ((z\x)*z).
 23 [para_into,10.1.1.2,8.1.1] (x*y)*z= (x* (z/x))* (y*x).
 26 [copy,21,flip.1] (x*y)* ((x\z)*x)=z* (y*x).
 36 [para_into,11.1.1.2,8.1.1] (x*x)*y= (x* (y/z))* (x*z).
 37 [para_into,11.1.1.2,2.1.1] (x*x)*y= (x*z)* (x* (z\y)).
 44 [para_from,11.1.1,6.1.1.1] ((x*y)* (x*z))/ (y*z)=x*x.
 46 [para_from,11.1.1,4.1.1.2] (x*x)\ ((x*y)* (x*z))=y*z.
 92 [para_into,21.1.1,8.1.1,flip.1](x*y)* ((x\ (z/ (y*x)))*x)=z.
 234 [para_into,26.1.1.1,2.1.1] x* ((y\z)*y)=z* ((y\x)*y).
 326,325 [para_into,36.1.1,11.1.1,flip.1]
    (x*((y*z)/u))*(x*u)=(x*y)*(x*z).
 404 [para_into,37.1.1,2.1.1,flip.1] (x*y)*(x*(y\((x*x)\z)))=z.
 768 [para_into,44.1.1.1.1,2.1.1] (x* (y*z))/ ((y\x)*z)=y*y.
 883,882 [para_into,46.1.1.2.2,2.1.1] (x*x)\ ((x*y)*z)=y*(x\z).
 884 [para_into,46.1.1.2,23.1.1,demod,883] ((x*y)/x)*(x\(z*x))=z*y.
 1614 [para_from,92.1.1,4.1.1.2,flip.1] (x\(y/(z*x)))*x=(x*z)\y.
 2321 [para_into,234.1.1,36.1.1,demod,326]
      (x*(y\z))*(x*y)=z*((y\(x*x))*y).
 3588 [para_from,404.1.1,4.1.1.2] (x*y)\z=x* (y\ ((x*x)\z)).
 3610 [copy,3588,flip.1] x* (y\ ((x*x)\z))= (x*y)\z.
 4362 [para_into,768.1.1.1.2,8.1.1] (x*y)/ (((y/z)\x)*z)=(y/z)*(y/z).
 4364 [para_into,768.1.1.1.2,2.1.1](x*y)/ ((z\x)*(z\y))=z*z.
 4898 [para_into,884.1.1.1.1,8.1.1] (x/(x/y))*((x/y)\(z*(x/y)))=z*y.
 9835 [para_from,4364.1.1,16.1.1.1] (x*x)\(y*z)=(x\y)*(x\z).
 15608,15607 [para_from,9835.1.1,3610.1.1.2.2,flip.1]
      (x*y)\(z*u)=x*(y\((x\z)*(x\u))).
 16487 [para_into,4362.1.1.2.1,16.1.1] (x*x)/(y*y)=(x/y)*(x/y).
 16490,16489 [para_from,16487.1.1,1614.1.1.1.2,demod,15608,3]
      (x\((y/x)*(y/x)))*x=(x\y)*(x\y).
 16511 [para_from,4898.1.1,2321.1.1.1,demod,9,16490,17,17,flip.1]
      (x*(y/z))*(z*z)=(x*z)*y.
 16515 [para_into,16511.1.1.1.2,6.1.1] (x*y)*(z*z)=(x*z)*(y*z).
 16516 [binary,16515.1,12.1] $F.
------------ end of proof -------------

Search stopped by max_proofs option.

============ end of search ============
```

10. Possibilities

As should be clear by now, our Otter proofs tend to be rather long, especially by comparison to many of the Otter proofs in other areas of algebra and logic; see, for instance, the much shorter proofs in [20]. In fact, while our *translated* proofs are hopefully concise and insightful, some of the actual *Otter* proofs we've generated are Titanic in length. The grand champion in this regard is a punishing proof about central nilpotency and Moufang loops that is over 300 printed pages long, with a formal length of proof over 2500. It took Otter nearly 2 days to find it.

All of this suggests that there is an intriguing depth to many of these results, and more importantly, betokens the huge potential in using Otter to help reveal ever deeper insights into the theory of quasigroups and loops.

Acknowledgements. I am enormously indebted to Ken Kunen for generously and patiently teaching me Otter, and to Michael Kinyon for learning it with me. I thank them both. Many of the results mentioned herein are joint work with one or both of them. I also thank the referee for his or her thorough and helpful comments. The paper is greatly improved because of them.

References

[1] **A. S. Basarab**: *Osborn's G-loops*, Quasigroups and Related Systems **1** (1994), 51 – 56.

[2] **V.D. Belousov**: *Foundations of the Theory of Quasigroups and Loops*, Izdat Nauka, Moscow 1967.

[3] **R. H. Bruck**: *A Survey of Binary Systems*, Springer-Verlag 1971.

[4] **O. Chein, H. Pflugfelder and J. D. H. Smith**: *Quasigroups and Loops, Theory and Applications*, Helderman-Verlag 1990.

[5] **S. Doro**: *Simple Moufang loops*, Math. Proc. of Camb. Phil. Soc. **83** (1978), 377 – 392.

[6] **T. Kepka, M. K. Kinyon and J. D. Phillips**: *F-quasigroups are isotopic to Moufang loops*, (in preparation).

[7] **M. K. Kinyon, K. Kunen and J. D. Phillips**: *Every diassociative A-loop is Moufang*, Proc. Amer. Math. Soc. **130** (2002), 619Ñ624, (electronic).

[8] **M. K. Kinyon, K. Kunen and J. D. Phillips**: *A generalization of Moufang and Steiner loops*, Algebra Universalis **48** (2002), 81 – 101.

[9] **M. K. Kinyon, K. Kunen and J. D. Phillips**: *Diassociativity in conjugacy closed loops*, Communications in Algebra, (to appear).

[10] **M. K. Kinyon, K. Kunen and J. D. Phillips**: *Loops with isotopically invariant flexibility*, (submitted).

[11] **M. K. Kinyon and J. D. Phillips**: *Commutants of Bol loops of odd order*, Proc. Amer. Math. Soc., (to appear).

[12] **M. K. Kinyon and J. D. Phillips**: *A note on trimedial quasigroups*, Quasigroups and Related Systems **9** (2002), 65 – 66.

[13] **M. K. Kinyon and J.D. Phillips**: *Moufang commutants*, (submitted).

[14] **K. Kunen**: *Moufang quasigroups*, J. Algebra **183** (1996), 231 – 234.

[15] **K. Kunen**: *Single axioms for groups*, J. Automated Reasoning **9(3)** (1992), 291 – 308.

[16] **K. Kunen**: *G-loops and permutation groups*, J. Algebra **220** (1999), 694 – 708.

[17] **W. W. McCune**: *OTTER 3.0 Reference Manual and Guide*, Technical Report ANL-94/6, Argonne National Laboratory, 1994; or see `http://www-fp.mcs.anl.gov/division/software`

[18] **W. W. McCune**: *Solution of the Robbins Problem*, J. Automated Reasoning **19(3)** (1997), 263 – 276.

[19] **W. W. McCune**: `http://www-unix.mcs.anl.gov/AR/new_results/`.

[20] **W. W. McCune and R. Padmanabhan**: *Automated Deduction in Equational Logic and Cubic Curves*, Springer-Verlag 1996.

[21] **H. Pflugfelder**: *Quasigroups and Loops, Introduction*, Helderman-Verlag 1990.

[22] **P. Syrbu**: *On loops with universal elasticity*, Quasigroups and Related Systems **3** (1996), 41 – 54.

[23] **L. Wos**: *Automated reasoning answers open questions*, Notices of the AMS **5(1)** (1993), 15 – 26.

Department of Mathematics & Computer Sciences
Wabash College
Crawfordsville, IN 47933
USA
e-mail: philipj@ wabash.edu

Received May 30, 2003