Generating Loops with the Inverse Property

John Slaney, Asif Ali
Australian National University, Australia

Abstract

This is an investigation in the tradition of Fujita *et al* (IJCAI 1993), Zhang *et al* (JSC 1996), Dubois and Dequen (CP 2001) in which CP or SAT techniques are used to answer existence questions concerning small algebras. In this paper, we open the attack on IP loops, an interesting and under-investigated variety intermediate between loops and groups.

1 Introduction

Automated reasoning techniques, particularly those of propositional satisfiability (SAT) and finite domain (FD) constraint satisfaction, are obviously applicable to the problem of enumerating small algebraic structures, and so should standardly be used to answer existence questions at least concerning very small cases. In this paper, we report on an investigation of IP loops, a variety intermediate between loops and groups which has been known for many years but to our knowledge never explored in much detail. We used FINDER [Sla94] to enumerate all IP loops up to order 13, and the commutative ones of order 14, and obtained a number of new results prompted by observation of these algebras. The numbers of algebras were reported in a recent note [AS08] and the full list of small IP loops is freely available online [SA07].

1.1 Algebraic background

A *quasigroup* is a groupoid with left and right division operators $/$ and $\backslash$. That is, it satisfies the laws:

$$x(x\backslash y) = y$$
$$(x/y)y = x$$
$$x\backslash xy = y$$
$$xy/y = x$$

In the finite case, this amounts simply to satisfying the left and right cancellation laws:

$$xy = xz \Rightarrow y = z$$
$$xz = yz \Rightarrow x = y$$

That is, its "multiplication table" is a Latin square, each row and each column being a permutation of the elements. A quasigroup is a *loop* iff it has a (right and left) identity: an element e such that

$$ex = x = xe$$

for all x. Loops in general are so numerous that almost all work on them has concerned special cases. One of the earliest classes of loops to be investigated was that of *Steiner loops*, which satisfy the additional postulates

$$(xy)y = x$$
$$x(xy) = y$$

Clearly, in any loop, each element x has a left inverse—an element y such that $yx = e$—and a right inverse

Sutcliffe G., Colton S., Schulz S. (eds.); Proceedings of ESARM 2008, pp. 55-66

z such that $xz = e$. In the case of Steiner loops, both y and z are just x. A weaker condition, also satisfied by groups, is that the left and right inverse operations coincide, meaning that for every x there is an element x^{-1} such that

$$xx^{-1} = e = x^{-1}x$$

Note that $(x^{-1})^{-1} = x$.

A loop is said to have the *inverse property*, and is called an *IP loop*, iff it is a loop with inverse such that for all elements x and y

$$x^{-1}(xy) = y = (yx)x^{-1}$$

It is not hard to see that IP loops also satisfy the principle $(xy)^{-1} = y^{-1}x^{-1}$. A Steiner loop is an IP loop of exponent 2 (i.e. such that $x^2 = e$ for all x) and a group is simply an associative IP loop. Moufang loops, which have been studied intensively, are IP loops satisfying the identity

$$x(z(yz)) = ((xz)y)z$$

IP loops are of interest as a strong and natural generalisation of both groups and Steiner loops. Moreover, they correspond exactly to semiassociative relation algebras [Mad82] in the same sense that groups correspond to (associative) relation algebras.[1] It is therefore a little surprising that they have attracted comparatively slight attention from algebraists.

The smallest IP loop that is not a group is of order 7:

$*$	1	2	3	4	5	6	7
$e=1$	1	2	3	4	5	6	7
2	2	3	1	6	7	5	4
3	3	1	2	7	6	4	5
4	4	7	6	5	1	2	3
5	5	6	7	1	4	3	2
6	6	4	5	3	2	7	1
7	7	5	4	2	3	1	6

x	x^{-1}
1	1
2	3
3	2
4	5
5	4
6	7
7	6

This structure has proper subalgebras $\{1,2,3\}$, $\{1,4,5\}$ and $\{1,6,7\}$. Note that the order of these subloops does not divide the order of the loop, marking a significant difference between IP loops and groups.[2] Associativity fails in that, for instance, $(2*2)*4 = 3*4 = 7$ while $2*(2*4) = 2*6 = 5$.

2 Generating IP loops

It is frequently useful to enumerate small examples of a class of algebraic structures, so that by examining what exists, and observing places where no such structures exist, the mathematician can gain a "feel" for the objects in question. At the simplest, the spectrum (the set of numbers n for which such algebras of order n exist) can have its initial segment settled by enumeration. In some cases, this suffices to allow the entire spectrum to be determined; in others, it merely dispposes of some awkward questions and suggests a conjecture concerning the rest. In many cases, "off the shelf" reasoning systems suffice for the enumeration, making this an attractive application domain for automated reasoning.

[1] Let $G = \langle S, * \rangle$ be a groupoid. The field of sets consisting of the power set of L, with $*$ raised to sets in the obvious pointwise manner, is a relation algebra iff G is a group, and a semiassociative relation algebra iff G is an IP loop.

[2] A loop in which the order of every subloop divides the order of the loop is said to have the *weak Lagrange property*. It has the *strong Lagrange property* if every subloop has the weak property.

```
solve satisfy;
int N;
type element = 1..N;

array[element,element] of var element: star;

constraint
  forall (x,y in element) (star[star[star[y,x],y],y] = x);

N = 3;
```

Figure 1: Zinc encoding of quasigroup existence problem QG5(3)

2.1 History

Fujita *et al* [FSB93] used the ICOT group's 'Model Generation Theorem Prover', a propositional reasoner in the style of SATCHMO [MB88], and other tools including FINDER to solve open problems in the theory of quasigroups by proving the existence or nonexistence of quasigroup models of certain equations. During the 1990s, this work was taken up and extended, notably by Hantao Zhang and his collaborators through the SATO system [ZBH96] and by McCune, Stickel and others [McC, ZS00]. In the constraint programming community, there were interesting developments concerning efficient encodings [DD01] and in the SAT community concerning symmetry avoidance [Zha96, AH01]. Recently, it has been shown [APSS05] that preprocessing of SAT encodings using restricted variants of resolution can simplify some of the quasigroup existence problems to the point that stochastic local search (SLS) solvers can successfully prove existence (though not nonexistence, of course). Meanwhile, the related problem of quasigroup completion [GS97] has become a well-established benchmark constraint satisfaction problem, offering as it does a nice balance between the highly structured and the random. Benchmark collections of SAT, CSP and SMT problems now routinely contain problems about quasigroups.

2.2 Problem representation

The simplest way to represent existence problems about quasigroups, loops, groups or other groupoids for automated reasoning purposes is to cast them as finite domain CSPs where each entry $\langle x, y\rangle$ in the "multiplication table" of the groupoid is a CSP variable whose domain consists of the elements of the algebra. Take for example the problem QG5(3). This requires the matrix

*	1	2	3
1			
2			
3			

to be filled with nine entries chosen from the values $1\ldots3$, in such a way that they form a Latin square and that the equation $(yx.y)y = x$ holds for all x and y. In fact, if they satisfy the equation, the cancellation properties follow. In the CSP modelling language Zinc [dlBMRW06] for instance, this is directly expressible (see Figure 1). Other such languages for constraint programming make it similarly easy to state the problem.

The equation flattens to $\forall x\forall y\forall w\forall z((yx = w \wedge wy = z) \Rightarrow zy = x)$ which has 4 variables and therefore $3^4 = 81$ domain-grounded instances obtained by substituting the three possible values 1, 2, 3 for the variables. Each of those instances relates a triple (possibly with repetition) of entries in the table,

and correspondingly imposes a constraint of cardinality at most 3 on the variables of the CSP. In the straightforward SAT recension, each possible value assignment $a * b = c$ is represented by a propositional variable p_{abc}, and each domain-grounded instance of the flattened equation becomes a 3-clause on these variables. Other encodings are possible, of course, but the suggested one is standard. Once the problem is so encoded, any FD or SAT solver can be used to solve it. Theorem provers, whether based on resolution and its variants or on term rewriting, can also be used to make inferences on either the first order or propositional levels.

To generate IP loops, we need another array of decision variables representing the inverse function, and of course the appropriate equations. It is useful to add a few redundant constraints, to strengthen propagation. We added the fact that inverse is of period 2 and the duality equation $(xy)^{-1} = y^{-1}x^{-1}$. Since we wish to enumerate isomorphism classes, it is important to avoid generating too many isomorphic copies of the solutions, which means we need to break symmetries. In order to break some of the many symmetries in a simple way, we required the identity e to be the lowest-valued element and x^{-1} to be in the range $x-1 \ldots x+1$, with self-inverse elements coming first in the order.

2.3 FINDER is good enough

For our work, we used FINDER, which stands somewhere between the FD and SAT solver families. It represents the problem in the FD manner rather than explicitly rendering it into SAT, so for example its variable selection heuristic looks at the FD variables, not at specific values for them—typically it looks for the smallest available domain—but it reasons somewhat like a SAT solver rather than in typical FD style. In particular, it uses unit resolution on the ground constraints together with forward-checking as its notion of local consistency, and it learns nogoods.

FINDER is far from representing the state of the art in finite model building: we expect that similar results could be produced faster using more recent technology such as Paradox, which is based on the much more efficient SAT solver Minisat. It suffices for our purpose, however, as it can find the solutions faster than they can be checked for isomorphism and has completed the order 13 search in reasonable time.

To count the isomorphism classes, it is necessary either to reason in a sophisticated way about symmetries during the search[3] or to remove redundant solutions from the output in a postprocessing phase. We chose the latter: our postprocessor takes each generated IP loop in turn and tries to generate from it an isomorphic copy that comes earlier in the (row-major) lexicographic order. If it succeeds, the generated loop is discarded; if it fails, the loop in question is the canonical one of its class and is output. This isomorphism removal method is rather slow, but requires little memory. Indicated future research includes incorporating isomorphism detection into the search.

Generating the IP loops of orders up to 11 is easy. We confirmed our results by obtaining the same numbers with MACE [McC]. Order 12 caused more difficulties, taking unreasonably long for both FINDER and MACE with their default settings. With a small change to make FINDER more aggressive about deleting old nogoods, however, we were able to solve the order 12 problem in a matter of hours.

Order 13 was more challenging. Our first partially successful run took over a week without exhausting the search space. On closer examination, we found that almost all of this time was taken up by the postprocessor eliminating isomorphic copies. We therefore somewhat strengthened the symmetry-breaking constraints, in order to reduce the number of copies to be treated, and rewrote the postprocessor to search less naïvely for dominating copies. The result is that the order 13 problem can now be completed in less than a day on a fairly ordinary desktop machine.

[3]The GAP-ECLiPSe hybrid of Gent *et al* [GHKL03] does this, and, given a suitably efficient underlying solver, may be the preferred method if the present investigations are to be pressed beyond order 13.

	Basic		Enhanced	
size	time (sec)	solutions	time (sec)	solutions
7	0.00	10	0.00	4
8	0.03	128	0.01	50
9	0.11	488	0.01	64
10	2.51	8856	0.50	1294
11	39.30	128488	1.25	5008
12	3026.31	8956032	231.38	626888

Figure 2: Basic *versus* enhanced symmetry breaking: FINDER runtimes and numbers of solutions before postprocessing. The fewer (redundant) solutions the better.

The most signficant part of the speedup was that due to the extra symmetry breaking constraints added by hand to the encoding. These reduced the domains of possible values for the cells in the second row of the table of the loop operation—the first row is fixed as it lists the elements of the form $e * x$, which of course is x in every case. The canonical representative of each isomorphism class is first in the lexicographic order in which this second row is most significant, so clearly we lose nothing by constraining the numbers early in the row to be as low as possible. Since e is the first (lowest-numbered) element, we can conveniently represent all elements as $(e+x)$ where x is an integer in the range $0 \ldots N-1$.[4] We are concerned to add constraints limiting the values of elements of the form $(e+1)*(e+x)$ where $0 < x < N$.

Where N is odd, this is simple. There are no fixed points for the inverse operation, so $(e+1)^{-1} = (e+2)$, so there are two possibilities for the value of $(e+1)*(e+1)$: it could be $(e+2)$ or it could be something else, where "something else" might as well be $(e+3)$ since all choices are symmetric. By similar reasoning, for $x > 1$, the canonical member of each isomorphism class has $(e+1)*(e+x) < (e+2x)$.

Where N is even, there is an additional complication. It is possible that some elements other than e are fixed points for inverse, and it can happen that for all of these fixed points a, the element $(e+1)*a$ is not a fixed point. In that case, the usual upper bound does not apply, but instead the values of such $(e+1)*a$ can be assigned arbitrarily. We choose to assign them in ascending order. We introduce a boolean flag (another decision variable) which will be set just in case the first 6 elements are all fixed points for inverse and $(e+1)*(e+2)$ is not a fixed point. Provided the flag is not set, a constraint similar to that for odd values of N applies. That is, using $\mathbf{f}(\varphi)$ to abbreviate $(e+1)*(e+\varphi)$:

Basic symmetry breakers:

$$e \leq x$$
$$x^{-1} < (x+2)$$
$$(x^{-1} = x \wedge y < x) \Rightarrow y^{-1} = y$$

For odd values of N:

$$x^{-1} < x+2$$
$$x^{-1} = x \Leftrightarrow x = e$$
$$\mathbf{f}(1) < (e+4)$$
$$(x > 1 \wedge 2x < N) \Rightarrow \mathbf{f}(x) < (e+2x)$$

[4]Naturally, we could let the elements *be* the integers $0 \ldots N-1$ or $1 \ldots N$, as in the Zinc encoding suggested in Figure 1, in which case the notation would be simplified. FINDER, however, is picky about types and complains if we confuse "element" with "int", so we keep the long-winded version for present purposes.

size	*quasigroups*	*loops*	*IPloops*	*groups*
1	1	1	1	1
2	1	1	1	1
3	5	1	1	1
4	35	2	2	2
5	1411	6	1	1
6	1130531	109	2	2
7	1.21×10^{10}	23746	2	1
8	2.70×10^{15}	1.06×10^{8}	8	5
9	1.52×10^{22}	9.37×10^{12}	7	2
10	2.75×10^{30}	2.09×10^{19}	47	2
11	— ? —	— ? —	49	1
12	— ? —	— ? —	2684	5
13	— ? —	— ? —	10342	1

Table 1: Numbers of algebras of given order

For even values of N:

$$\mathbf{f}(1) = e$$
$$(\neg\text{FLAG} \wedge 0 < x < N/2) \Rightarrow \mathbf{f}(x) < (e + 2x + 1)$$
$$\text{FLAG} \Rightarrow (e+5)^{-1} = (e+5)$$
$$(\text{FLAG} \wedge x > 1 \wedge (e+x)^{-1} = (e+x)) \Rightarrow (\mathbf{f}(x))^{-1} \neq \mathbf{f}(x)$$
$$(\text{FLAG} \wedge 1 < x < y \wedge (e+y)^{-1} = (e+y)) \Rightarrow \mathbf{f}(x) < \mathbf{f}(y)$$

The enhanced symmetry breaking pays handsomely, as shown in Figure 2 where the runtimes and numbers of solutions (before postprocessing) with and without the extra symmetry breakers are compared. It is worth noting that in generating 626,888 solutions to the order 12 problem, for example, FINDER backtracks only 202,549 times. That means that three quarters of the branches in the search tree end in solutions, so it is unlikely that significant improvement to the efficiency of the search is possible. Any future advance will need to involve better symmetry removal, to cut down still further the number of solutions generated.

2.4 The numbers

Table 1 shows the count of all IP loops of small orders. For comparison, the number of these which are associative (i.e. groups) is also shown, as are the numbers of quasigroups and loops.[5] Clearly, at very small sizes, associativity has no room to fail given the loop and inverse postulates. Up to order 4, indeed, the existence of an identity element alone is enough to force a quasigroup to be an abelian group. It seems from the table, however, that IP loops will resemble loops rather than groups in that, once clear of the initial noise, the numbers of such algebras will show a monotonic exponential increase with size. At the same time, it can be seen that the existence of a two-sided inverse is in some intuitive sense a "strong" property: of the 10^{30} quasigroups of order 10, one in every hundred billion is a loop, but of these loops only one in every 40 million trillion has an inverse.

An important subvariety of any variety of groupoid is that obtained by imposing a postulate of commutativity. The numbers of commutative IP loops are shown in Table 2. It was something of a surprise

[5]The numbers of quasigroups and loops are taken from the paper of McKay *et al* [MMM07] which also contains an account of the many errors making up the history of counting these objects.

size	*groups*	*non − groups*	*total*
1	1		1
2	1		1
3	1		1
4	2		2
5	1		1
6	1		1
7	1		1
8	3		3
9	2		2
10	1	5	6
11	1	1	2
12	2	12	14
13	1	7	8
14	1	179	180

Table 2: Numbers of Commutative IP loops of given order

to observe that the smallest such loop which is not a group is of order 10. It seems that commutativity tends to enforce associativity, at least at small sizes: only one of the 48 non-associative IP loops of order 11, for instance, is commutative.

3 New results

In abstract algebra, the effect of generating the structures of small sizes is often to provide a supply of data rather than a supply of theorems. This means that model searches function more like experiments in an empirical science than like proof searches in mathematics as standardly conceived. The rôle of diagrams in traditional geometry is somewhat similar: a diagram is not a proof, but it can supply a disproof, and inspection of diagrams can suggest conjectures to the mathematician with an eye for regularities. In the same way, identifying patterns in the numbers or distribution of small structures is a good way of formulating conjectures in abstract algebra.

In the present case, we have been able to use the "data" provided by FINDER to arrive at several new results concerning IP loops. These are not necessarily very deep mathematics, and their proofs, once the regularities have been observed, are not especially hard. The trick is to formulate the conjecture in the first place, and for this purpose access to the quasi-empirical data is invaluable.

3.1 Order of subloops

Steiner loops satisfy the condition $\forall x(x^2 = e)$ or equivalently $\forall x(x^{-1} = x)$. We wondered whether there was anything to say about the distribution of elements satisfying the self-inverse condition in IP loops which are not Steiner loops in general. Fortunately, in generating the algebras, as explained in §2 above, part of our technique was to set the inverse operation before generating the loop operation. Thus we were presented immediately with the numbers of IP loops of each order with each possible choice of inverse, where the difference between two inverse operations is just in the number of self-inverse elements. Hence our generation method itself resulted in a study of the distribution of self-inverse elements among IP loops of each size. To our initial surprise, there appeared to be no such elements at all (other than the

size	$k=2$	$k=3$	$k=4$	$k=5$
2	1		1	
3		1		
4	1		2	
5				1
6				
7		1		
8	1		4	
9		2		
10	1		10	
11				
12				
13		64		10

Table 3: Numbers of IP loops satisfying $x^k = e$

identity) in IP loops of odd order. We knew, of course, that Steiner loops are always of even order, but expected that IP loops of any cardinality would typically contain at least *some* fixed points for inverse.

They do not, however, as can be shown by a simple counting argument:

Theorem 1. *Let $L = \langle S, * \rangle$ be a finite IP loop. Then the cardinality of S is even iff L has an element of order 2—that is, an element a such that $a \neq e$ but $a^2 = e$.*

Proof. Left to right, the result is trivial: since the inverse operation is of period 2, the set of elements of L which are *not* fixed points for it must be of even cardinality. If the order of L is even, therefore, there must also be an even number of self-inverse elements, so e cannot be the only such element.

For the converse, suppose a is self-inverse and distinct from e. Let the operation L_a be defined on S by the equation $L_a(x) = a * x$. Then L_a is of period 2, as $L_a(L_a(x)) = a * (a * x) = a^{-1} * (a * x) = x$. Moreover, L_a has no fixed point, as if $L_a(x) = x$ then $a * x = x$ so $a = e$ contrary to the supposition of the theorem. Therefore L_a partitions S into pairs, so $|S|$ is even. □

Corollary 2. *No IP loop of odd order has a subloop of even order.*

3.2 IP loops of exponent k

Following on from this theorem, we examined the spectra of the equations $x^k = e$ for small values of k. The observations up to order 13 are summarised in Table 3.[6]

The spectrum of Steiner loops (the column $k = 2$ in the table) is well known to consist of 1 and all integers congruent to 2 or 4 (mod 6). The argument that all Steiner loops fall into that spectrum is nice enough to be worth rehearsing here. First note that Steiner loops are commutative, because for any elements x and y, $(x.xy)(yx) = y(yx) = x = (x.xy)(xy)$ so $xy = yx$. Next, if $xy = z$ then $xz = x(xy) = y$, so such loops are "fully commutative" in that for any triple x, y and z, the six equations obtained by permuting the variables in "$xy = z$" are all equivalent. Steiner loops thus correspond directly to Steiner triple systems, or sets of triples of elements from a set such that every pair of elements from the set occurs in exactly one triple. The identity of the loop is added to allow for the case where x and y are the same. Evidently, there are three pairs in every triple, so the number of triples in a Steiner triple system is one

[6] As usual, we assume association to the left, defining $x^0 = e$ and $x^{k+1} = x^k * x$.

third of the number of pairs of elements in the set. Thus, where the set has n elements, $n^2 - n$ must be divisible by 6. Expressing n as $6k+i$ for some i in 0..5, we see immediately that $i^2 - i$ must be divisible by 6, requiring i to be either 0, 1, 3 or 4. Hence $n+1$, the order of the Steiner loop, must be congruent to 1, 2, 4 or 5 (mod 6). But odd orders greater than 1 are impossible by Theorem 1, so except for the degenerate case $n = 1$ the order of the loop must be congruent to 2 or 4 (mod 6).

The core of this argument, divisibility by 6, can be generalised.

Theorem 3. *Let L be an IP loop of order $3n$. Then L contains an element x distinct from e such that $x^2 = x^{-1}$.*

Proof. Consider any elements a, b and c, all distinct from e, such that $ab = c$ in L. Then the following all hold:

$ab = c$
$cb^{-1} = a$
$a^{-1}c = b$
$b^{-1}a^{-1} = c^{-1}$
$bc^{-1} = a^{-1}$
$c^{-1}a = b^{-1}$

Moreover, the six table entries represented by these equations are all distinct unless one of them is of the form $xx = x^{-1}$. If L contains no such x, therefore, the table entries not involving e are partitioned into blocks of 6. There are $(3n-1)^2 - (3n-1)$ such entries, so $(3n-1)^2 - (3n-1)$ is a multiple of 6. That is, $9n^2 - 9n + 2$ is a multiple of 6. Let $3n = 6k+i$ where $0 \leq i < 6$. Then $9(6k+i)^2 - 9(6k+i) + 2$ is divisible by 6, so $9(36k + 12ki + i^2) - 56k - 9i + 2$ is divisible by 6, so $9(i^2 - i) + 2$ is divisible by 6. But it is not. □

Theorem 4. *Let L be an IP loop of exponent 5. Let n be the order of L. Then either $n \equiv 1$ (mod 12) or $n \equiv 5$ (mod 12).*

Proof. For any element x of L, $x^4 = x^{-1}$ and it is not hard to show that $x^3 = (x^2)^{-1}$. It is left as a satisfying exercise to show that for all x, i and j, $x^i * x^j = x^{i+j}$ (mod 5). It follows that L is composed of a number of subloops of order 5, each of course of the form $\{e, x, x^2, x^3, x^4\}$. That is, they are disjoint except for e. Therefore $n \equiv 1$ (mod 4).

Clearly, L contains no element x distinct from e such that $x^2 = x^{-1}$, so by Theorem 3 its order is not a multiple of 3 and therefore $n \not\equiv 9$ (mod 12). □

Theorem 5. *Let L be an IP loop of exponent 3. Let n be the order of L. Then either $n \equiv 1$ (mod 6) or $n \equiv 3$ (mod 6).*

Proof. For every element x of L, $x^3 = e$ or equivalently $x^2 = x^{-1}$. It follows that n is odd, and also that Theorem 3 is not directly useful. We can adapt the argument, however. If we ignore the first row and column of the table, we are left with $(n-1)^2$ entries. Each row contains two "anomalous" entries: e in the x^{-1} column and x^{-1} on the diagonal. Removing those two entries from each row, that leaves $(n-1)^2 - 2(n-1)$ to be filled with blocks of 6 as before. Thus $(n-1)^2 - 2(n-1)$ is divisible by 6. Expressing n as $6k+i$, we find that $i^2 - 4i + 3$ is divisible by 6, which is to say $i = 1$ or $i = 3$. □

3.3 The square property

A groupoid has the *square property* iff $(xy)^2 = x^2y^2$ for all x and y. It is well known that a group is commutative iff it has the square property. This is not true of IP loops, however. The smallest counterexample

is of order 10:

$*$	1	2	3	4	5	6	7	8	9	10
$e=1$	1	2	3	4	5	6	7	8	9	10
2	2	1	4	3	6	5	9	10	7	8
3	3	4	1	2	7	8	5	6	10	9
4	4	3	2	1	9	10	8	7	5	6
5	5	6	7	9	2	1	10	3	8	4
6	6	5	8	10	1	2	3	9	4	7
7	7	9	5	8	10	3	4	1	6	2
8	8	10	6	7	3	9	1	4	2	5
9	9	7	10	5	8	4	6	2	3	1
10	10	8	9	6	4	7	2	5	1	3

x	x^{-1}
1	1
2	2
3	3
4	4
5	6
6	5
7	8
8	7
9	10
10	9

This IP loop is commutative, but lacks the square property as $(3*5)^2 = 4$ but $3^2 * 5^2 = 2$. The converse is also not valid for IP loops: there are 3 non-commutative IP loops of order 12 with the square property, and 2 more of order 13.

Another property which suffices for a group to be abelian is that it is of order p^2 where p is a prime. IP loops of such orders are not in general commutative, as for example there are 5 non-commutative ones of order 9. However, both commutative IP loops of order 9 are groups, leading us to wonder whether all commutative IP loops of order p^2 are groups. The answer is negative: there is a commutative non-associative IP loop of order 11, so its direct product with any IP loop also of order 11 is a non-commutative IP loop of order 121 which is not a group.

3.4 Some rare IP loops

A loop is said to be *flexible* iff it satisfies

$$x(yx) = (xy)x$$

and *alternative* iff

$$x(xy) = (xx)y$$
$$(xy)y = (xy)y$$

Steiner loops and groups are flexible and alternative. It turns out that the smallest IP loop which is flexible and alternative but neither a group nor a Steiner loop is of order 12, and there are, up to isomorphism, only two such loops of order 12 and none of order 13.

Steiner loops and groups are also *C-loops*, meaning they satisfy

$$x(y(yz)) = ((xy)y)z$$

All C-loops are known to be IP loops and alternative. Up to order 13, there is only one non-associative, non-Steiner C-loop. Again it is of order 12.

4 Future directions

This paper has added to the store of known "small" examples of core algebraic structures. IP loops inhabit the space between very tightly constrained varieties (groups, Steiner loops) and very loose ones (quasigroups). They are closely related to an interesting generalisation of relation algebras. We have detailed the IP loops up to the orders at which the number becomes too big for a mathematician to know them all. The most obvious extensions of our work are:

1. Complete the account of the spectrum of IP loops of exponent k, for all k. We have the impression that it is not very difficult, but settling this issue properly would be satisfying.

2. Extend the investigation to particular classes of IP loops. For example, enumerate the small C-loops. Since these are comparatively rare, it will be necessary to go to larger sizes before enumeration ceases to be worthwhile. Phillips and Vojtěchovský [PV06] report very small numbers of C-loops up to order 14, for which they used MACE-4. It is possible that significantly extending the search may raise different challenges for automated reasoning.

3. Investigate the use of GAP-ECLiPSe or a similar hybrid which brings computational group theory to bear on the problem of symmetries in search spaces. Since this detects many symmetries and avoids them early, it is potentially an important tool for getting further with the enumeration of IP loops or species of them.

4. Experiment with more systematic symmetry breakers such as the "least number" heuristic of Jian Zhang and its extensions. Dealing with symmetry, rather than with search inefficiency, is the main bottleneck in the algebra generation process at present.

5. Pick out some new benchmark problems from our work, for finite domain constraint solvers, for SAT solvers or for SMT systems.[7]

References

[AH01] Gilles Audemard and Laurent Henocque. The extended least number heuristic. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR)*, pages 427–442, 2001.

[APSS05] Anbulagan, Duc Nghia Pham, John K. Slaney, and Abdul Sattar. Old resolution meets modern SLS. In *Proceedings of the National Conference of the American Association for Artificial Intelligence (AAAI)*, pages 354–359, 2005.

[AS08] Asif Ali and John Slaney. Counting loops with the inverse property. *Quasigroups and Related Structures*, 16:13–16, 2008.

[DD01] Gilles Dequen and Olivier Dubois. The non-existence of a (3,1,2)-conjugate orthogonal Latin square of order 10. In *Principles and Practice of Constraint Programming (CP)*, pages 108–120, 2001.

[dlBMRW06] Maria Garcia de la Banda, Kim Marriott, Reza Rafeh, and Mark Wallace. The modelling language Zinc. In *Principles and Practice of Constraint Programming (CP)*, pages 700–705, 2006.

[FSB93] Masayuki Fujita, John Slaney, and Frank Bennett. Automatic generation of some results in finite algebra. In *Proceedings of the thirteenth International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 52–57, 1993.

[GHKL03] Ian Gent, Warwick Harvey, Tom Kelsey, and Steve Linton. Generic SBDD using computational group theory. In *Principles and Practice of Constraint Programming (CP)*, pages 333–347, 2003.

[GS97] Carla Gomes and Bart Selman. Problem structure in the presence of perturbation. In *Proceedings of the National Conference of the American Association for Artificial Intelligence (AAAI)*, pages 221–226, 1997.

[Mad82] Roger Maddux. Some varieties containing relation algebras. *Transaction of the American Math Society*, 272:501–526, 1982.

[MB88] Rainer Manthey and François Bry. SATCHMO: A theorem prover implemented in Prolog. In *Proceedings of the ninth Conference on Automated Deduction (CADE-12)*, pages 415–434, 1988.

[7]This research was supported by NICTA (National ICT Australia) and by the Australian National University. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. The authors wish to acknowledge useful discussions with colleagues Tomasz Kowalski and Brendan McKay, and the constructive comments of the anonymous referees.

[McC] William McCune. Prover9 and MACE 4. http://www.cs.unm.edu/∼mccune/mace4/.

[MMM07] Brendan McKay, Alison Meynert, and Wendy Myrvold. Small latin squares, quasigroups and loops. *Journal of Combinatorial Designs*, 15:98–119, 2007.

[PV06] J. D. Phillips and Petr Vojtěchovský. C-loops: An introduction. *Publicationes Mathematicae Debrecen*, 68:115–137, 2006.

[SA07] John Slaney and Asif Ali. IP loops of small order, 2007. http://users.rsise.anu.edu.au/∼jks/IPloops/.

[Sla94] John Slaney. FINDER, finite domain enumerator: System description. In *Proceedings of the twelfth Conference on Automated Deduction (CADE-12)*, pages 798–801, 1994.

[ZBH96] Hantao Zhang, Maria Paola Bonacina, and Jieh Hsiang. PSATO: a distributed propositional prover and its application to quasigroup problems. *Journal of Symbolic Computation*, 11:1–18, 1996.

[Zha96] Jian Zhang. Constructing finite algebras with FALCON. *Journal of Automated Reasoning*, 17:1–22, 1996.

[ZS00] Hantao Zhang and Mark Stickel. Implementing the Davis-Putnam method. *Journal of Automated Reasoning*, 24:277–296, 2000.