Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Symbolic Computation 41 (2006) 811–817
Journal of Symbolic Computation
www.elsevier.com/locate/jsc

Symbolic computation with finite quandles

Richard Henderson[a], Todd Macedo[b], Sam Nelson[b,*]

[a] *Red Hat, Inc., 444 Castro Street, Suite 1200, Mountain View, CA 94041, United States*
[b] *Department of Mathematics, University of California, Riverside, 900 University Avenue, Riverside, CA 92521, United States*

Received 23 August 2005; accepted 21 March 2006
Available online 2 May 2006

Abstract

Algorithms are described and *Maple* implementations are provided for finding all quandles of order n, as well as computing all homomorphisms between two finite quandles or from a finitely presented quandle (e.g., a knot quandle) to a finite quandle, computing the automorphism group of a finite quandle, etc. Several of these programs work for arbitrary binary operation tables and hence algebraic structures other than quandles. We also include a stand-alone C program which finds quandles of order n and provide URLs for files containing the results for $n = 6$, 7 and 8.

Keywords: Virtual knot invariants; Finite quandles; Symbolic computation

1. Introduction

In 1980, David Joyce introduced a new algebraic structure dubbed the *quandle*. Quandles are tailor-made for defining invariants of knots since the quandle axioms are essentially the Reidemeister moves written in algebra. Associated to any knot diagram, there is a quandle called the *knot quandle* which is a complete invariant of knot type up to homeomorphism of topological pairs.

The history of quandle theory is a story of rediscovery and reinvention. Quandles and their generalization, racks, have been independently invented and studied by numerous authors (Brieskorn, 1988; Fenn and Rourke, 1992; Joyce, 1982; Mateev, 1982, etc.) and

* Corresponding author. Tel.: +1 951 827 3113; fax: +1 951 827 7314.
E-mail addresses: rth@redhat.com (R. Henderson), tmace001@student.ucr.edu (T. Macedo), knots@esotericka.org (S. Nelson).

0747-7171/$ - see front matter
doi:10.1016/j.jsc.2006.03.002

classification results for various subcategories of quandles have been obtained by various authors (Graña, 2004; Nelson, 2003). In Ho and Nelson (2005), the third listed author and a coauthor described a way of representing finite quandles as matrices and implemented algorithms for finding all finite quandles, removing isomorphic quandles from the list, and computing the automorphism group of each quandle. As we later learned, some of our work has duplicated the efforts of others (Ryder, 1992; Lopes and Roseman, 0000; Carter et al., 2004).

This paper is intended to reduce future duplication of effort by describing the algorithms for computation with finite quandles implemented in Ho and Nelson (2005) and other recent projects, as well as an improved algorithm for finding quandle matrices. The C source for our implementation of this algorithm as well as *Maple* implementations of algorithms for computing with finite quandles and the lists of quandle matrices of order 6, 7 and 8 are available for download at http://www.esotericka.org/quandles. Additional Maple code corresponding to current and future projects will be made available at the same site, such as an algorithm for finding all Alexander presentations of a quandle when such exist (Murillo et al., 0000).

2. Quandles, quandle matrices, and homomorphisms

Definition 1. A *quandle* is a set Q with a binary operation $\triangleright : Q \times Q \to Q$ satisfying

(i) for every $x \in Q$ we have $x \triangleright x = x$,
(ii) for every $x, y \in Q$ there is a unique $z \in Q$ such that $x = z \triangleright y$, and
(iii) for every $x, y, z \in Q$ we have $(x \triangleright y) \triangleright z = (x \triangleright z) \triangleright (y \triangleright z)$.

If $(Q, \triangleright)$ satisfies (ii) and (iii), Q is a *rack*.

Axiom (ii) says that $\triangleright$ is right-invertible; for every $y \in Q$, the map $f_y : Q \to Q$ defined by $f_y(x) = x \triangleright y$ is a bijection (indeed, a quandle automorphism). Denote the inverse map as $f_y^{-1}(x) = x \triangleleft y$. Then $(Q, \triangleleft)$ is also a quandle, called the *dual* of $(Q, \triangleright)$; not only is $\triangleleft$ self-distributive, but it is an easy exercise to check that $\triangleright$ and $\triangleleft$ distribute over each other.

Standard examples of quandles include groups, which are quandles under conjugation $g \triangleright h = h^{-1}gh$ as well as n-fold conjugation $g \triangleright h = h^{-n}gh^n$, denoted as $\mathrm{Conj}(G)$ and $\mathrm{Conj}_n(G)$ respectively, and Alexander quandles, which are modules over the ring $\Lambda = \mathbb{Z}[t^{\pm 1}]$ of Laurent polynomials in one variable with integer coefficients, with quandle operation given by

$$x \triangleright y = tx + (1-t)y.$$

A finite quandle Q may be specified by giving its *quandle matrix* M_Q, which is the matrix obtained from the operation table of $Q = \{x_1, x_2, \ldots, x_n\}$ (where the entry in row i column j is $x_i \triangleright x_j$) by dropping the xs and keeping only the subscripts. In Ho and Nelson (2005) it is noted that, unlike arbitrary binary operation tables or indeed even rack tables, quandle axiom (i) permits us to deduce the column and row labels from the elements along the diagonal of a quandle matrix.

Example 1. Let $Q = R_4$, the dihedral quandle of order 4, which has underlying set $Q = \{x_1 = 0, x_2 = 1, x_3 = 2, x_4 = 3\}$ with quandle operation $x_i \triangleright x_j = x_{2j-i \pmod 4}$. Then Q has operation table

	x_1	x_2	x_3	x_4
x_1	x_1	x_3	x_1	x_3
x_2	x_4	x_2	x_4	x_2
x_3	x_3	x_1	x_3	x_1
x_4	x_2	x_4	x_2	x_4

and hence matrix $M_{R_4} = \begin{bmatrix} 1 & 3 & 1 & 3 \\ 4 & 2 & 4 & 2 \\ 3 & 1 & 3 & 1 \\ 2 & 4 & 2 & 4 \end{bmatrix}$.

A map $\phi : Q \to Q'$ from a quandle $Q = \{x_1, \dots, x_n\}$ to a quandle $Q' = \{y_1, \dots, y_m\}$ may be represented by a vector $v = (\phi(x_1), \phi(x_2), \dots, \phi(x_n)) \in Q'^n$. Such a vector v then represents a homomorphism iff $\phi(x_i \triangleright x_j) = \phi(x_i) \triangleright \phi(x_j)$, that is, iff we have

$$v[A[i, j]] = B[v[i], v[j]]$$

for all $x_i, x_j \in Q$ where $A = M_Q$, $B = M_{Q'}$, and the notation $M[i, j]$ indicates the entry of M in row i column j.

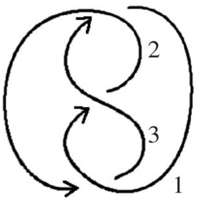
In Fenn and Rourke (1992), presentations of quandles by generators and relations are defined. In Nelson (2005), it is observed that all finitely presented quandles may be written with a *short form* presentation in which every relation is of the form $a = b \diamond c$ where $\diamond \in \{\triangleright, \triangleleft\}$. In particular, a knot quandle has a presentation with n such short relations where n is the number of crossings in the diagram. Moreover, we may assume (rewriting if necessary) that every relation is written in the form $a = b \triangleright c$ and that no two relations of the form $a = b \triangleright c$ and $a' = b \triangleright c$ are present, since if $a = b \triangleright c$ and $a' = b \triangleright c$ are both present we can replace every instance of a' with a and remove the generator a' without changing the presented quandle; in particular, if our quandle is a knot quandle, Reidemeister type I moves[1] induce such a replacement.

Definition 2. Let $Q = \langle 1, 2, \dots, n \mid a_1 = b_1 \triangleright c_1, \dots, a_m = b_m \triangleright c_m, m \le n^2 \rangle$ be a short form quandle presentation such that no two relations of the form $a_i = b_i \triangleright c_i$ and $a_j = b_i \triangleright c_i$ with $a_i \neq a_j$ are present. The matrix $MP \in M_n(\mathbb{Z})$ with

$$MP[i, j] = \begin{cases} k & k = i \triangleright j \text{ a listed relation} \\ 0 & \text{otherwise} \end{cases}$$

is the *matrix of the presentation Q*. Note that a quandle matrix for a finite quandle is the matrix of a presentation of a finite quandle, so this definition generalizes the notion of quandle matrices to finitely presentable quandles.

Example 2.



$$MP_{KQ} = \begin{bmatrix} 0 & 3 & 0 \\ 0 & 0 & 2 \\ 1 & 0 & 0 \end{bmatrix}$$

The pictured trefoil knot diagram has quandle presentation $\langle 1, 2, 3 \mid 1 = 2 \triangleright 3, 2 = 3 \triangleright 1, 3 = 1 \triangleright 2 \rangle$. The relations are determined at a crossing by looking in the positive direction of the overcrossing strand indicated by the given orientation; the relation is

(left-hand undercrossing) = (right-hand undercrossing) $\triangleright$ (overcrossing).

See Fenn and Rourke (1992) or Nelson (2005) for more.

This matrix representation gives us a convenient way to do computations involving quandles, including the quandle counting invariant for knot quandles or other short form quandles with respect to a finite target quandle. The next section describes algorithms for doing computations with quandles and refers to implementations in *Maple* (Nelson, 0000) and C (Henderson, 0000).

[1] Reidemeister moves are described in Nelson (2005) and many other works.

3. Algorithms

The goal of the computations in Ho and Nelson (2005) was to find all quandles of a given finite order. Originally, we wrote separate programs for each value of n; Nelson (0000) includes one example of such an implementation, `quandleslist5`. We later wrote a more general program which works for arbitrary n, though due to the large number of columns to be checked, for values of $n \geq 6$ we decided to implement a stand-alone version suitable for distributed computing.

The algorithm implemented in `quandleslist` takes a number n and generates a list of all $n \times n$ standard form quandle matrices. A matrix $M \in M_n(\mathbb{Z})$ is a quandle matrix in standard form iff it satisfies the following three conditions:

(i) for $i \in \{1, \dots, n\}$, $M[i, i] = i$,
(ii) every column in M is a permutation of $\{1, \dots, n\}$, and
(iii) for every triple $1 \leq i, j, k \leq n$ we have $M[M[i, j], k] = M[M[i, k], M[j, k]]$.

To guarantee that conditions (i) and (ii) are satisfied, we start by getting a list of all permutations of $\{1, \dots, n\}$. The program `listperms` takes a number n and produces a list of all permutations $\rho \in \Sigma_n$, represented as vectors $[\rho(1), \rho(2), \dots, \rho(n)]$, in the dictionary order.

The ith column in a standard form quandle matrix has entry i in the ith position. The program `listpermsi` takes a pair of positive integers (n, i) and outputs a list of all permutations of $\{1, 2, \dots, n\}$ $\rho \in \Sigma_n$ which fix the element i.

To test quandle axiom (iii), we note that the first time any triple (i, j, k) fails to satisfy the axiom, we can exit the program and report that the matrix is not a quandle. This is implemented in `q3test`.

For a fixed value of n, we can then simply run over a series of nested loops, testing each resulting matrix for quandle axiom (iii), since by construction axioms (i) and (ii) are already satisfied. The program `quandleslist5` is an example of this.

The program `quandleslist` finds a list of all quandle matrices of a given size n. To find all $n \times n$ quandle matrices for arbitrary n, `quandleslist` finds all control vectors $v[i]$ with n entries using `listmaps`, a program which takes two inputs a and b and outputs a list of all a-component vectors with entries in $\{1, \dots, b\}$. Each entry in the control vector corresponds to a column in the output matrix; for each such control vector, an $n \times n$ matrix $M[i, j]$ is produced whose ith column is $L[n, i][v[i]]$, where $L[n, i]$ is the output of `listpermsi(n,i)`. These matrices are then tested for quandle axiom (iii) using `q3test`. For completeness, we include a program which tests a matrix for all three quandle axioms, `qtest`.

Since every n-component vector with entries in $\{1, \dots, m\}$ can be interpreted as a map from $\{1, \dots, n\}$ to $\{1, \dots, m\}$, we can use `listmaps` to compute the set of all homomorphisms from one finite quandle to another. Let $A \in M_n(\mathbb{Z})$ be an $n \times n$ quandle matrix and $B \in M_m(\mathbb{Z})$ an $m \times m$ quandle matrix. Then the vector $v \in \mathbb{Z}^n$, $1 \leq v[i] \leq m$, represents a quandle homomorphism $v : A \to B$ iff

$$v[A[i, j]] = v(i \triangleright j) = v(i) \triangleright v(j) = B[v[i], v[j]],$$

as noted in Section 2. The program `homtest` takes two quandle matrices and a vector and reports whether the vector represents a quandle homomorphism or not.

The program `homtest` handles the case where A is either a finite quandle matrix or a presentation matrix for a finitely presented quandle; in the former case, the program simply tests whether the assignment of generators $\{1, \dots, n\}$ in the quandle with presentation matrix A

to elements $\{1, \dots, m\}$ in the finite quandle B satisfy the relations defining A by ignoring any zero entries in A.

We make use of `nextmap`, a procedure which takes as input a vector v and number n and returns the next m-component vector with entries in $\{1, 2, \dots, m\}$ in the dictionary order, to get a list of all homomorphisms from the quandle with matrix A to the quandle with matrix B in the program `homlist`. The program `homcount` counts the number of homomorphisms from one finite quandle to another. If A is a knot quandle presentation matrix, then `homcount` computes the quandle counting invariant, i.e., the number of quandle colorings of the knot diagram defining A by the finite quandle B. Alternate methods of computing the quandle counting invariant for finite Alexander quandles are described in Dionísio and Lopes (2003).

After the first version of this paper was completed, we implemented a much faster algorithm for finding quandle homomorphisms, `homlist2`. This program uses a $|B|$-component vector with entries in $\{0, 1, \dots, |A|\}$ as a template for a homomorphism, with 0 entries acting as blanks to be filled in. The program keeps a working list of such templates, systematically filling in zero values and propagating the value through the template using `homfill`. The procedure `homfill` takes as input a quandle matrix B, a quandle presentation matrix A and a template vector v and systematically checks every pair of entries for the quandle homomorphism condition $v[A[i, j]] = B[v[i], v[j]]$, filling in zeros where possible and reporting "false" if a contradiction is found.

Since an isomorphism is a bijective homomorphism, and a bijective map is represented by a permutation $v : \{1, \dots, n\} \to \{1, \dots, n\}$, we can test whether two quandles given by matrices are isomorphic by running through the list of permutations of order n and testing to see whether any are homomorphisms. The program `isotest` returns "true" the first time it finds an isomorphism and "false" if it gets through all $n!$ permutations without finding an isomorphism.[2]

Replacing `listmaps` in `homlist2` with `permute(n)` and setting $B = A$ gives us the automorphism group of the quandle with matrix A, `autlist`, represented as a list of permutation vectors.

In Ho and Nelson (2005), a slightly different method of determining the automorphism group of a quandle was used. Specifically, permuting the entries of a quandle matrix A by a permutation ρ applies an isomorphism to the defined quandle, but the new matrix now has its rows and columns out of order. To restore the order, we conjugate by the matrix of the permutation; the resulting matrix was called $\rho(A)$ in Ho and Nelson (2005). In particular, a permutation ρ is an automorphism of A iff $\rho(A) = A$. To compute $\mathrm{Aut}(A)$ in Ho and Nelson (2005), we ran a loop over the list of permutations given by `listperms` and noted which ones preserved the original matrix A. Here, we include a program `stndiso` which computes the standard form matrix $\rho(A)$ given a quandle matrix A and a vector v representing the permutation ρ.

Finally, once we have a list of quandle matrices of order n, we want to choose a single representative from each isomorphism class. The program `reducelist` takes a list of quandle matrices and compares them pairwise with `isotest`, removing isomorphic copies and outputting a trimmed list. The program `reducelist` works for short lists; an improved algorithm, implemented as `reducelist2`, is better for longer lists, but neither is sufficient to reduce the rather lengthy lists of quandles of order 7 and 8 in a reasonable amount of time.

We note that several of these programs, notably `homtest`, `homlist`, `homlist2`, `homcount`, `isotest`, `autlist`, and `reducelist`, are not quandle-specific but apply as written to any binary

[2] A faster version of this program using orbit decompositions of finite quandles is described in Nelson and Wong (0000).

Table 1
Number of complete matrices tested

n	5	6	7	8
Total search space	8.0×10^6	3.0×10^{11}	1.0×10^{20}	4.1×10^{30}
With early testing	8400	715680	1.0×10^8	n/a
With forward propagation	1154	53500	5.0×10^6	7.7×10^8
Total quandles	404	6658	152900	5225916

Table 2
Number of tests pruned with early testing

Column	2	3	4	5	6
$n = 5$	186	8736	14626	–	–
$n = 6$	4728	1090404	8418374	1187556	–
$n = 7$	154680	2.3×10^8	1.8×10^{10}	3.8×10^9	1.8×10^8

Table 3
Number of times columns constrained beyond axiom (i)

Column	3	4	5	6	7
$n = 5$	164	179	290	–	–
$n = 6$	3558	4396	3348	5020	–
$n = 7$	115872	228384	91452	82910	117430

operation defined using a matrix as an operation table. These facts are exploited in Murillo et al. (0000), in which the authors give a program which determines all Alexander structures on a quandle, if there are any, using matrices to represent the Cayley table of an abelian group.

We have also implemented a stand-alone version of `quandleslist`, written in C (see Henderson (0000)); it writes a list of quandle matrices in *Maple* format to an output file.

In our initial version of the stand-alone program, several instances of the program could be run in parallel on networked machines using a control file to ensure that separate instances do not repeat the same computations. However, sufficient improvements were made to the algorithm by pruning the search space that the current version can handle the $n = 8$ case on a single processor, though the $n = 9$ case is still out of reach even with a large network.

The first improvement was to introduce a partial test versus axiom (iii) after generation of each column. In many cases we can find entries that violate the axiom well before the entire matrix is generated, which allows vast portions of the search space to be pruned.

The second improvement was to notice when all of the interior coordinate values as well as the left-hand side value of the axiom (iii) equality have been computed, but the right-hand side value has not. In this case we can constrain a row of a future column to be equal to the left-hand side value. This reduces the number of rows that must be permuted when searching that column, which further prunes the search space. The earlier these constraints are added, the more the pruning effect is magnified. For example, with $n = 7$ and all else held equal, adding a single constraint to column 3 saves $(6! - 5!) * ((6!)^4)$ or 2.3×10^{16} tests, whereas adding a constraint to column 7 saves only $6! - 5!$ or 600 tests.

The effect of the two improvements can be seen in Tables 1–3. It is interesting to note that although there is nothing in the program to prevent it (and reasonable amount of code to encourage it), we never add constraints to column 2, nor do we ever add more than one constraint per column, or detect addition of conflicting constraints.

References

Brieskorn, E., 1988. Automorphic sets and braids and singularities. Contemp. Math. 78, 45–115.
Carter, S., Kamada, S., Saito, M., 2004. Surfaces in 4-space. In: Low-Dimensional Topology, III. In: Encyclopaedia of Mathematical Sciences, vol. 142. Springer-Verlag, Berlin.
Dionísio, F.M., Lopes, P., 2003. Quandles at finite temperatures. II. J. Knot Theory Ramifications 12, 1041–1092.
Fenn, R., Rourke, C., 1992. Racks and links in codimension two. J. Knot Theory Ramifications 1, 343–406.
Graña, M., 2004. Indecomposable racks of order p^2. Beiträge Algebra Geom. 45, 665–676.
Henderson, R. http://www.esotericka.org/quandle5.c.
Ho, B., Nelson, S., 2005. Matrices and finite quandles. Homology Homotopy Appl. 7, 197–208.
Joyce, D., 1982. A classifying invariant of knots, the knot quandle. J. Pure Appl. Algebra 23, 37–65.
Lopes, P., Roseman, D. On finite racks and quandles, arXiv.org: math.GT/0412487.
Mateev, S.V., 1982. Distributive groupoids in knot theory. Mat. Sb. 119, 78–88.
Murillo, G., Nelson, S., Thompson, A. Matrices and finite Alexander quandles, arXiv.org: math.GT/0509365.
Nelson, S. http://www.esotericka.org/quandles.
Nelson, S., 2003. Classification of finite Alexander quandles. Topology Proc. 27, 245–258.
Nelson, S., 2005. Signed ordered knotlike quandle presentations. Algebr. Geom. Topol. 5, 443–462.
Nelson, S., Wong, C.-Y. On the orbit decomposition of finite quandles, arXiv.org: math.GT/0508247.
Ryder, H., 1992. The structure of racks. Ph.D. Thesis, U. Warwick.