

AN ENUMERATION PROCESS FOR RACKS

JIM HOSTE AND PATRICK D. SHANAHAN

ABSTRACT. Given a presentation for a rack \mathcal{R} , we define a process which systematically enumerates the elements of \mathcal{R} . The process is modeled on the systematic enumeration of cosets first given by Todd and Coxeter. This generalizes and improves the diagramming method for n -quandles introduced by Winker. We provide pseudocode that is similar to that given by Holt, Eick, and O’Brien for the Todd-Coxeter process. We prove that the process terminates if and only if \mathcal{R} is finite, in which case, the procedure outputs an operation table for the finite rack. We conclude with an application to knot theory.

1. INTRODUCTION

The fundamental quandle of an oriented knot or link is an algebraic invariant which was proven to be a complete invariant of knots (up to mirror reversal) by Joyce [8]. See also Matveev [9]. While it is easy to find a presentation of the quandle of a link using a modification of the Wirtinger algorithm, it is usually difficult to determine the quandle’s isomorphism class. A more tractable, but less sensitive, invariant is the n -quandle of a link which is a certain quotient of the fundamental quandle.

In his PhD thesis [12], Winker introduced a method to produce a Cayley diagram of the n -quandle of a link. His diagramming method is a graph-theoretic modification of a fundamental process in computational group theory called the Todd-Coxeter process [10]. This process was introduced to find the index of a finitely generated subgroup H in a finitely presented group G . In addition, the process produces a table which describes the right action of G on the set of cosets of H . The process is incorporated in many computer algebra systems.

Sarah Yoseph made a preliminary investigation of a Todd-Coxeter like process for the enumeration of n -quandles in her (unpublished) undergraduate senior thesis directed by the second author. Her work complemented Winker’s by considering a table-based approach to n -quandle enumeration and producing elementary pseudocode. In this paper, we apply the table-based approach to the more general structure of a rack. Our development of an enumeration process for a rack \mathcal{R} given by a presentation $\langle S \mid R \rangle$ will be modeled on the exposition of the Todd-Coxeter process given in Holt-Eick-O’Brien [3]. The rack enumeration process we present extends Winker’s work to the study of racks and provides pseudocode for its implementation.

An important feature of the currently accepted Todd-Coxeter process is that if the index of H is finite, then the process will terminate in a finite number of steps. In [11], Ward showed that this was not true of the original process and provided

Received by the editor July 12, 2017 and, in revised form, February 16, 2018.

2010 *Mathematics Subject Classification*. Primary 20-04; Secondary 57M25.

a modification to the process to eliminate this problem. Using arguments similar to those in [3], we prove that if our rack enumeration process completes, then the resulting output is rack isomorphic to \mathcal{R} and, moreover, that the process completes if and only if \mathcal{R} is finite. We also provide an example demonstrating the importance of Ward's modification in the rack setting as well.

In the special case of quandles, the Todd-Coxeter process could be used in theory to determine the structure of any finite quandle. This is because Joyce proved that every quandle Q is isomorphic to a quandle structure on the set of cosets of a particular subgroup of the automorphism group of Q . However, employing this approach would require determining a presentation for $\text{Aut}(Q)$ and generators for the appropriate subgroups which may not be practical. In the case of knot and link quandles, Joyce also proved that the coset quandle of the peripheral subgroups of the fundamental group is isomorphic to the fundamental quandle of the link. The authors extend this result to n -quandles of links in [5]. Hence, the Todd-Coxeter process can be used to investigate the structure of the n -quandle of a link, giving an alternative to Winker's method. Given these theoretical and practical limitations, it is desirable to have an enumeration procedure which applies directly to any finitely presented rack.

In Section 2 we review the basic definitions of racks and rack presentations. We introduce enumeration tables and the rack enumeration process in Section 3. We prove that the tables produced satisfy five basic properties which are used later to prove the main result in Section 4. We also include pseudocode for the processes introduced in this section. Finally, in Section 4, we prove that if the process completes, then the output is isomorphic to the rack and, moreover, that a finitely presented rack is finite if and only if the process completes. In Section 5, we provide an example showing the importance of Ward's modification in the rack setting and discuss an alternative modification. We conclude with an application to knot theory.

2. RACKS AND PRESENTATIONS

We begin with the definition and some basic properties of racks. Excellent sources for this material are [2], [1], [7], [8], and [12].

Definition 2.1. A set \mathcal{R} with two binary operations \triangleright and \triangleright^{-1} is a **rack** if the following two properties hold:

- R1.** $(x \triangleright y) \triangleright^{-1} y = (x \triangleright^{-1} y) \triangleright y = x$ for all $x, y \in \mathcal{R}$, and
- R2.** $(x \triangleright y) \triangleright z = (x \triangleright z) \triangleright (y \triangleright z)$ for all $x, y, z \in \mathcal{R}$.

Properties R1 and R2 are sometimes referred to as the right cancellation and right self-distributive axioms, respectively. It is easy to show that R1 and R2 imply $(x \triangleright^\epsilon y) \triangleright^\delta z = (x \triangleright^\delta z) \triangleright^\epsilon (y \triangleright^\delta z)$ for $\epsilon, \delta \in \{-1, 1\}$. In general, a rack is non-associative and the following well-known lemma can be used to rewrite any product as a left-associated product.

Lemma 2.2. Let \mathcal{R} be a rack and $x, y, z \in \mathcal{R}$. If $\epsilon, \delta \in \{-1, 1\}$, then $x \triangleright^\epsilon (y \triangleright^\delta z) = ((x \triangleright^{-\delta} z) \triangleright^\epsilon y) \triangleright^\delta z$.

Proof. Using the cancellation and one of the distributive properties we have:

$$x \triangleright^\epsilon (y \triangleright^\delta z) = ((x \triangleright^{-\delta} z) \triangleright^\delta z) \triangleright^\epsilon (y \triangleright^\delta z) = ((x \triangleright^{-\delta} z) \triangleright^\epsilon y) \triangleright^\delta z.$$

□

A convenient notation introduced by Fenn and Rourke in [2] uses Lemma 2.2 to avoid the use of parentheses. From this point on, we shall adopt Fenn and Rourke's **exponential notation** defined by

$$x^y = x \triangleright y \quad \text{and} \quad x^{\bar{y}} = x \triangleright^{-1} y.$$

With this notation, x^{yz} will represent $(x^y)^z = (x \triangleright y) \triangleright z$, whereas, by Lemma 2.2, $x^{\bar{y}z}$ will be used to represent $x^{(y^z)} = x \triangleright (y \triangleright z)$.

Given an integer m , we will also let x^{y^m} denote $x^{y \cdots y}$ if $m > 0$, x if $m = 0$, and $x^{\bar{y} \cdots \bar{y}}$ if $m < 0$, where in each case there are $|m|$ factors of y or \bar{y} in the exponent.

Definition 2.3. A rack Q is called a **quandle** if $x \triangleright x = x$ for all $x \in Q$. Further, if $n \geq 2$ is an integer, then a quandle Q is called an n -**quandle** if $x^{y^n} = x$ for all $x, y \in Q$.

Notice that in an n -quandle we also have that $x^{\bar{y}^n} = x$. A 2-quandle is also called an **involutory quandle**.

Following Fenn and Rourke, we define a presentation $\langle S | R \rangle$ of a rack with generating set S and relations R as a quotient of a free rack. For any set S , let $F(S)$ denote the free group on S and in this group let \bar{w} represent the inverse of the element w .

Definition 2.4. The **free rack** on S is the set of equivalence classes

$$FR(S) = \{[a^w] \mid a \in S, w \in F(S)\},$$

where $[a^u] = [b^v]$ if $a = b$ in S and $u = v$ in $F(S)$. The operations in $FR(S)$ are defined by $[a^u] \triangleright [b^v] = [a^{u \bar{v} b v}]$ and $[a^u] \triangleright^{-1} [b^v] = [a^{u \bar{v} \bar{b} v}]$.

From this point on, we will abuse notation and simply let a^u represent the equivalence class $[a^u]$.

A **congruence** on a rack \mathcal{R} is an equivalence relation \sim that respects the operations. In particular, if $\mathcal{R} = FR(S)$, then a congruence is a relation with the property that if $a^s \sim b^t$ and $x^u \sim y^v$, then $a^{s \bar{u} x u} \sim b^{t \bar{v} y v}$ and $a^{s \bar{u} \bar{x} u} \sim b^{t \bar{v} \bar{y} v}$. Given a congruence on $FR(S)$, then the congruence classes form a quotient of $FR(S)$ that is itself a rack. This notion of a quotient rack allows us to define a rack in terms of generators and relations.

Let S be a finite set of generators and let R be a finite set of relations in $FR(S)$. That is, R is a finite set of ordered pairs of the form (a^u, b) where $a, b \in S$ and $u \in F(S)$. More formally,

$$R = \{(a_i^{u_i}, b_i) \mid a_i, b_i \in S, u_i \in F(S), 1 \leq i \leq r\} \subseteq FR(S) \times FR(S).$$

The rack given by the presentation $\langle S | R \rangle$ is then defined to be the quotient of $FR(S)$ by the smallest congruence \sim_R containing R . The smallest congruence is described more concretely by Fenn and Rourke in terms of *consequences* of the relations in R . Using their work we can derive the following proposition.

Proposition 2.5. If $\mathcal{R} = \langle S | R \rangle$, then $x^s \sim_R y^t$ if and only if x^s can be taken to y^t by a finite sequence of the following substitutions or their inverses. For all $a, b, c \in S$ and $u, v, w \in F(S)$:

- (1) Replace a^{uw} with $a^{uv\bar{v}w}$.
- (2) If $(a^u, b) \in R$, then replace a^{uw} with b^w .
- (3) If $(a^u, b) \in R$, then replace c^{vw} with either $c^{v\bar{u}au\bar{b}w}$ or $c^{v\bar{u}\bar{a}ubw}$.

We shall refer to the substitutions in Proposition 2.5 as **substitution moves**. The proof of the proposition requires showing that the congruence defined by the substitution moves is the same as the congruence defined by consequences of the relations described in Fenn and Rourke. We leave the details to the interested reader.

Remark 2.6. Notice that since the word w is arbitrary in each of the substitution moves in Proposition 2.5 it follows that if $x^s \sim_R y^t$, then $x^{sw} \sim_R y^{tw}$ for any $w \in F(S)$.

As is customary with group presentations, we shall adopt the notation $a^u = b$ to represent a relation $(a^u, b) \in R$ and $x^s = y^t$ to denote $x^s \sim_R y^t$ in the rack $\langle S \mid R \rangle$. Notice that if all relations $a^a = a$ for $a \in S$ are included in R , then $\langle S \mid R \rangle$ is a quandle. Moreover, for a fixed n , if R additionally includes all relations $a^{b^n} = a$ for all distinct $a, b \in S$, then $\langle S \mid R \rangle$ is an n -quandle. To see that this is the case, first notice that if $a^{b^n} = a$, then it follows by substitution move (3) that $x^{b^n ab^n \bar{a}} = x$ for all x . Thus, we obtain $y^{ab^n} = y^{b^n a}$ for all y by considering $x = y^{b^n}$. Similarly, $y^{\bar{a}b^n} = y^{b^n \bar{a}}$. Since this is true for all generators, it follows by induction that $y^{wb^n} = y^{b^n w}$ for all words $w \in F(S)$. Now consider arbitrary elements $x = a^u$ and $y = b^v$. Since the relation $a^{b^n} = a$ has been added and, in the case $a = b$, $a^{a^n} = a$ since $\langle S \mid R \rangle$ is a quandle, we have

$$x^{y^n} = a^{u(\bar{v}bv)^n} = a^{u\bar{v}b^n v} = a^{b^n u\bar{v}v} = a^u = x.$$

3. THE RACK ENUMERATION PROCESS

In this section, we introduce the notion of an enumeration table and identify important properties of these tables that will remain unchanged during the enumeration process. Let $\langle S \mid R \rangle$ be a rack where $S = \{x_1, \dots, x_g\}$ and R is a set of relations $x_{i_k}^{u_k} = x_{j_k}$ with u_k a reduced word in $F(S)$ for $1 \leq k \leq r$ and $1 \leq i_k, j_k \leq g$. Let $\bar{S} = \{\bar{x}_1, \dots, \bar{x}_g\}$. Following Winker, we call the relations in R **primary relations**. Notice that for each primary relation $x_{i_k}^{u_k} = x_{j_k}$ and for any $x \in \langle S \mid R \rangle$ we have, by substitution move (3), that

$$x^{\bar{u}_k x_{i_k} u_k \bar{x}_{j_k}} = x.$$

These relations are called **secondary relations** by Winker. The word $\bar{u}_k x_{i_k} u_k \bar{x}_{j_k}$ may not be reduced, in which case, we will use its reduced form in the procedure. We denote the set of reduced secondary relations by R_2 .

Definition 3.1. An **enumeration table** \mathcal{E} for a rack $\mathcal{R} = \langle S \mid R \rangle$ is a 4-tuple (ω, A, τ, ρ) where ω is the number of rows in the table, A is a partial function from $\{1, 2, \dots, \omega\} \times (S \cup \bar{S})$ to $\{1, 2, \dots, \omega\}$, $\tau : \{1, 2, \dots, \omega\} \rightarrow \mathcal{R}$ is a function, and $\rho : \{1, 2, \dots, \omega\} \rightarrow \{1, 2, \dots, \omega\}$ is a function with the property that $\rho(i) \leq i$ for all $1 \leq i \leq \omega$.

We will denote $A(i, y)$ by i^y , and so i^y may or may not be defined since A is a partial function. Define the **live elements** of \mathcal{E} to be the set $\Omega = \{i \mid 1 \leq i \leq \omega \text{ and } \rho(i) = i\}$ and call \mathcal{E} **complete** if for every $i \in \Omega$ and for every $y \in S \cup \bar{S}$, we have that i^y is defined.

If \mathcal{R} is a finite rack, we describe a process that produces a sequence of tables $\mathcal{E}_0, \mathcal{E}_1, \dots, \mathcal{E}_f$ so that \mathcal{E}_f is complete, Ω is a rack with operations provided by \mathcal{E}_f , and $\tau : \Omega \rightarrow \mathcal{R}$ is an isomorphism. In our description of the process we will represent

an enumeration table by a rectangular array whose rows are numbered 1 through ω and whose columns are labelled by the elements of $S \cup \bar{S}$, τ , and ρ . The entry in row i and column $y \in S \cup \bar{S}$ is i^y if it is defined and empty otherwise. The last two columns give values of τ and ρ , respectively, for the row label i . We begin with an example that illustrates the process before giving the details of the algorithms involved.

Consider the rack with presentation

$$\mathcal{R} = \langle a, b \mid a^{ba} = b, b^{ba} = a, a^{bb} = a, b^{aa} = b \rangle.$$

We initialize the enumeration table \mathcal{E} by letting 1 represent the element a and 2 represent b . That is, we define $\tau(1) = a$ and $\tau(2) = b$ and we set $\rho(1) = 1$ and $\rho(2) = 2$. We next find the set R_2 of secondary relations. For each secondary relation $x^w = x$ we record the reduced word w .

$$R_2 = \{\bar{a}\bar{b}aba\bar{b}, \bar{a}\bar{b}, \bar{b}\bar{b}abb\bar{a}, \bar{a}\bar{a}baa\bar{b}\}.$$

\mathcal{E}	a	b	\bar{a}	\bar{b}	τ	ρ
1					<u>a</u>	1
2					<u>b</u>	2

The next step is to encode information from the primary relations. Consider the first relation $a^{ba} = b$. Since $\tau(1) = a$ and $\tau(2) = b$, we would like our table to satisfy $1^{ba} = 2$. However, 1^b is not defined in \mathcal{E} so we *define* a new element $3 = 1^b$ and extend the map τ so that $\tau(3) = \tau(1)^b = a^b$. Since $1^b = 3$ we also add the *inverse entry* $3^{\bar{b}} = 1$. We indicate where a definition is made by underlining the defined entry in the table.

Notice that $1^{ba} = 2$ and $1^b = 3$ imply that $3^a = 2$. This is called a *deduction* and we also encode it, and its inverse entry $2^{\bar{a}} = 3$, in our table. This is called *scanning* the first primary relation and the process is illustrated by a *helper table* shown to the right of the enumeration table below. Parentheses in the helper table indicate where *forward* and *backward scanning* end. The deduction $3^a = 2$ occurs where open parentheses meet. The corresponding entries added to \mathcal{E} are enclosed with open parentheses.

\mathcal{E}	a	b	\bar{a}	\bar{b}	τ	ρ		
1		<u>3</u>			<u>a</u>	1		
2			(3)		<u>b</u>	2		
3	(2)			<u>1</u>	<u>a^b</u>	3		

							<u>b</u>	<u>a</u>	
							1	3)	(2)

Scanning the remaining primary relations gives another definition and three additional deductions. Again we mark the modified entries in the table to indicate whether they came from a definition or a deduction and include the helper tables for the relations.

\mathcal{E}	a	b	\bar{a}	\bar{b}	τ	ρ		
1		3	(4)	(3)	<u>a</u>	1		
2	(3)	<u>4</u>	3		<u>b</u>	2		
3	2	(1)	(2)	1	<u>a^b</u>	3		
4	(1)			<u>2</u>	<u>b^b</u>	4		

							<u>b</u>	<u>a</u>	
							1	3)	(1)
							<u>b</u>	<u>b</u>	
							2)	(3	2)

The table \mathcal{E} above represents the conclusion of a definite loop in the rack enumeration process that scans all primary relations. The next loop in the process

scans each secondary relation for each live row. Since scanning may introduce new live rows, this loop is indefinite.

Consider scanning the first secondary relation $\bar{a}\bar{b}ab\bar{a}\bar{b}$ for live row 1. A helper table for this scan is shown below. Notice that scanning forward from 1 we have $1^{\bar{a}bab} = 1$ is defined but we cannot scan forward further because 1^a is not defined. So we begin scanning backwards from 1. In doing so we have that $1^{ba} = 2$ is defined and we have arrived at a *coincidence* where two different values, 1 and 2, appear in the same location in the helper table (which we denote by $[1_2]$).

	\bar{a}	\bar{b}	a	b	a	\bar{b}
1	4	2	3	$[1_2]$	3	1

To resolve this coincidence, we eliminate the larger index 2 and merge any data from row 2 into row 1 of the table. We do this by first changing the value of $\rho(2)$ to be 1, which indicates that 2 is a dead row and that all occurrences of 2 will eventually be replaced. Then for each $x \in \{a, b, \bar{a}, \bar{b}\}$ we do one of three things. If 2^x is undefined, we proceed to the next value for x . If $2^x = i$ and 1^x is undefined, then we remove $2^x = i$ and $i^{\bar{x}} = 2$ from \mathcal{E} and add $1^x = i$ and $i^{\bar{x}} = 1$ to \mathcal{E} . Notice this situation occurs for $x = a$. Otherwise, if $2^x = i$ and $1^x = j$, then we also remove $2^x = i$ and $i^{\bar{x}} = 2$ but, instead of adding new entries, we queue up a new coincidence between i and j . Notice this situation occurs for $x = b$ and $x = \bar{a}$ and, in both cases, the new coincidence is $[4_3]$. After resolving the coincidence $[4_3]$ in the same manner, no new coincidences appear and the resulting table is shown below. The entries in the table changed by the coincidences are marked by closed parentheses and rows 2 and 4 are now dead rows.

\mathcal{E}	a	b	\bar{a}	\bar{b}	τ	ρ
1	[3]	3	[3]	3	a	1
2	$\cancel{3}$	$\cancel{4}$	$\cancel{3}$		b	1
3	[1]	1	[1]	1	a^b	3
4	$\cancel{1}$		$\cancel{2}$	b^b		3

At this stage the enumeration table is complete but the process is not. Continuing to scan the secondary relations will never lead to a new definition or deduction, however, there could be additional coincidences. The reader can verify that all remaining scans *complete correctly*, that is, forward scanning reaches the end of the relation without any definitions, deductions, or coincidences needed. It now follows, as we show later, that the rack is of order 2 with $\mathcal{R} = \{a, a^b\}$. Moreover, a multiplication table for the rack can now be derived from the complete enumeration table using the rack axioms.

In Algorithm 1, we present pseudocode for the rack enumeration process described in the example. The pseudocode contains several subroutines that will be defined subsequently. Since the process contains an indefinite loop, a run limit is used to guarantee that the process terminates. We say that the ENUMERATE process **completes** when it returns a complete table in line 24. Specifically note that, if the process completes, then all secondary relations have been scanned for all live rows.

Algorithm 1 The rack enumeration process

```

1: procedure ENUMERATE( $S, R, M$ )
2:   Input: generators  $S$ , primary relations  $R$ , run limit  $M$ 
3:    $(R_2, \mathcal{T}) := \text{INIT}(S, R)$                                  $\triangleright$  derive  $R_2$ , initialize table
4:   for  $x_i^u = x_j \in R$  do                                      $\triangleright$  scan primary relations
5:     SCAN( $\sim \mathcal{T}$ , REP( $i$ ),  $u$ , REP( $j$ ))
6:   end for
7:    $i := 1$ 
8:   while  $i \leq \max(\Omega)$  and  $i \leq M$  do
9:     for  $w \in R_2$  do                                          $\triangleright$  scan secondary relations
10:    if  $i \in \Omega$  then
11:      SCAN( $\sim \mathcal{T}$ ,  $i, w, i$ )
12:    else
13:      break                                                  $\triangleright i$  is dead, stop scanning
14:    end if
15:  end for
16:  if  $i \in \Omega$  then
17:    for  $y \in S \cup \bar{S}$  and  $i^y$  undefined do
18:      DEFINE( $\sim \mathcal{T}, i, y$ )                                $\triangleright$  fill undefined entries in row  $i$ 
19:    end for
20:  end if
21:   $i := i + 1$ 
22: end while
23: if  $i > \max(\Omega)$  then
24:   return  $\mathcal{E}$                                           $\triangleright$  process completes
25: else
26:   return run limit exceeded
27: end if
28: end procedure

```

Before we describe the subroutines called by ENUMERATE, we list five properties which we will show to be true after the enumeration table is initialized and which remain true after each step of the procedure. These properties will then allow us to produce the rack isomorphism $\tau : \Omega \rightarrow \mathcal{R}$ when the process completes. First, we need some additional definitions. Let $w = y_1 y_2 \dots y_t \in F(S)$ and let $j \in \{1, 2, \dots, \omega\}$. We say j^w is **defined and equal** to k if $j_0 = j$ and for $1 \leq i \leq t$ we have $j_i = j_{i-1}^{y_i}$ is defined and $j_t = k$. As seen in the example, the function ρ will be used to record when coincidences occur. Let $\text{orbit}(i) = \{\rho^t(i) \mid t \geq 0\}$ where ρ^t is ρ composed with itself t times. Define the **least representative** of i by $\text{REP}(i) = \min(\text{orbit}(i))$.

Property 1. $1 \in \Omega$ and $\tau(i) = x_i$ for all $1 \leq i \leq g$.

Property 2. If $i, j \in \{1, 2, \dots, \omega\}$ and $y \in S \cup \bar{S}$, then $i^y = j$ if and only if $j^{\bar{y}} = i$.

Property 3. If $i, j \in \{1, 2, \dots, \omega\}$, $y \in S \cup \bar{S}$, and $i^y = j$, then $\tau(i)^y = \tau(j)$ in \mathcal{R} .

Property 4. If $j \in \Omega$, then there exists $i \in \Omega$, $1 \leq i \leq g$, and $w \in F(S)$ such that $j = i^w$.

Property 5. If $i \in \{1, 2, \dots, \omega\}$, then $\tau(i) = \tau(\text{REP}(i))$ in \mathcal{R} .

Notice that the single element $y \in S \cup \bar{S}$ in both Properties 2 and 3 can be replaced by any word $w \in F(S)$. This is easily proven by inducting on the length of w .

Next, we introduce Algorithms 2 and 3. The first initializes the enumeration table and produces a set of reduced secondary relations. The second creates a new row in the table and two new entries. The notation $\sim \mathcal{T}$ in the argument list of DEFINE (and already appearing in Algorithm 1) means that the procedure changes \mathcal{E} . We adopt this convention throughout.

Algorithm 2 Initializing the table

```

1: procedure INIT( $S, R$ )
2:   Input: generators  $S$ , primary relations  $R$ 
3:    $\omega := g$ ;  $A := \phi$ ;  $R_2 := \phi$ 
4:   for  $x_i^u = x_j \in R$  do                                 $\triangleright$  derive secondary relations  $R_2$ 
5:      $w := \bar{u}x_iu\bar{x}_j$  (reduced)
6:      $R_2 := R_2 \cup \{w\}$ 
7:   end for
8:   for  $1 \leq i \leq \omega$  do
9:      $\tau(i) := x_i$ 
10:     $\rho(i) := i$ 
11:   end for
12:    $\mathcal{T} := (\omega, A, \tau, \rho)$ 
13:   return  $(R_2, \mathcal{T})$ 
14: end procedure

```

Algorithm 3 Defining i^y

```

1: procedure DEFINE( $\sim \mathcal{T}, i, y$ )
2:   Input:  $\mathcal{T}$ ,  $i \in \Omega$ ,  $y \in S \cup \bar{S}$ 
3:    $\omega := \omega + 1$                                  $\triangleright$  add new row to table
4:    $i^y := \omega$ ;  $\omega^{\bar{y}} := i$ 
5:    $\tau(\omega) := \tau(i)^y$ 
6:    $\rho(\omega) := \omega$ 
7: end procedure

```

It is straightforward to show the following.

Proposition 3.2. *Properties 1–5 are true after calling INIT and they are preserved by each call to DEFINE.*

By saying a call to DEFINE preserves the properties, we mean that if they are true before a call to DEFINE, then they remain true after the call. We next define the procedure SCAN in Algorithm 4. It will call on DEFINE and the additional routines DEDUCTION and COINCIDENCE, that will be given in Algorithms 5 and 9, respectively.

Algorithm 4 Scanning the relation $i^w = j$

```

1: procedure SCAN( $\sim\mathcal{T}, i, w, j$ )
2:   Input:  $\mathcal{T}, i, j \in \Omega$ ,  $w = y_1y_2\dots y_t \in F(S)$ , reduced
3:    $f := 1$ ;  $b := t$ ; ▷ initialize forward and backward counters
4:    $k := i$ ;  $\ell := j$ ; ▷ initialize forward and backward scans
5:   while  $f \leq b$  do
6:     while  $f \leq b$  and  $k^{y_f}$  defined do ▷ scan forward
7:        $k := k^{y_f}$ ;  $f := f + 1$ 
8:     end while
9:     while  $f \leq b$  and  $\ell^{\bar{y}_b}$  defined do ▷ scan backward
10:       $\ell := \ell^{\bar{y}_b}$ ;  $b := b - 1$ 
11:    end while
12:    if  $f < b$  then
13:      DEFINE( $\sim\mathcal{T}, k, y_f$ ) ▷ extend forward scan
14:    else if  $f = b$  then
15:      DEDUCTION( $\sim\mathcal{T}, k, y_f, \ell$ ) ▷ scans meet
16:      break ▷ break from while loop
17:    else if  $k \neq \ell$  then ▷  $b < f$ 
18:      COINCIDENCE( $\sim\mathcal{T}, k, \ell$ ) ▷ scans overlap incorrectly
19:    else
20:      break ▷ scan completes correctly
21:    end if
22:  end while
23: end procedure

```

The SCAN procedure scans forward as far as possible and then scans backward as far as possible. After doing so, if there is a gap, then a definition is made and the cycle is repeated until the scans meet or overlap. This leads to a DEDUCTION or COINCIDENCE, respectively. Furthermore, it is not difficult to prove that because the word w is reduced, if DEFINE is called, then the procedure ends with a call to DEDUCTION.

Algorithm 5 Making the deduction $i^y = j$

```

1: procedure DEDUCTION( $\sim\mathcal{T}, i, y, j$ )
2:   Input:  $\mathcal{T}, i, j \in \Omega$ ,  $y \in S \cup \bar{S}$ 
3:    $i^y := j$ ;  $j^{\bar{y}} := i$ 
4: end procedure

```

In order to see that a call to SCAN preserves Properties 1–5, it suffices to show that each call to the subroutines DEDUCTION and COINCIDENCE preserves the properties.

Proposition 3.3. *Properties 1–5 are preserved by each call to DEDUCTION.*

Proof. Since no new rows are added and no values of τ and ρ are changed by DEDUCTION, Properties 1, 4, and 5 are clearly preserved. Property 2 is preserved since DEDUCTION adds both $i^y = j$ and $j^{\bar{y}} = i$ to the table.

We now discuss Property 3. We need only consider the case where k^y is undefined before the call to DEDUCTION and $k^y = l$ after the call. Suppose this occurred from a call to $\text{SCAN}(i, y_1 y_2 \dots y_t, j)$. Then, the $f = b$, $i^{y_1 \dots y_{f-1}} = k$, k^{y_f} is not defined, $j^{\bar{y}_t \dots \bar{y}_{b+1}} = \ell$, and $\ell^{\bar{y}_b}$ is not defined. The deduction adds two new entries $k^{y_f} = \ell$ and $\ell^{\bar{y}_f} = k$ to the table, so we must prove $\tau(k)^{y_f} = \tau(\ell)$ and $\tau(\ell)^{\bar{y}_f} = \tau(k)$.

Because Properties 1–5 were satisfied up to this call to DEDUCTION, we have that $\tau(k) = \tau(i)^{y_1 \dots y_{f-1}}$ and $\tau(\ell) = \tau(j)^{\bar{y}_t \dots \bar{y}_{f+1}}$. Now there are two cases depending on whether the scan was applied to a primary or secondary relation.

Case 1. Assume $1 \leq i, j \leq g$ and $x_i^{y_1 \dots y_t} = x_j$ is a primary relation. With the notation above, we have

$$\begin{aligned}
 \tau(k) &= \tau(i)^{y_1 \dots y_{f-1}} \\
 \tau(k)^{y_f} &= \tau(i)^{y_1 \dots y_f} && \text{Remark 2.6} \\
 &= x_i^{y_1 \dots y_f} \\
 &= x_i^{y_1 \dots y_f y_{f+1} \dots y_t \bar{y}_t \dots \bar{y}_{f+1}} && \text{from INIT}(S, R) \\
 &= x_j^{\bar{y}_t \dots \bar{y}_{f+1}} && \text{Prop. 2.5 (1)} \\
 &= \tau(j)^{\bar{y}_t \dots \bar{y}_{f+1}} && \text{Prop. 2.5 (2)} \\
 &= \tau(j)^{\bar{y}_t \dots \bar{y}_{f+1}} && \text{from INIT}(S, R) \\
 &= \tau(\ell).
 \end{aligned}$$

It follows from Remark 2.6 and Proposition 2.5 (1) that $\tau(\ell)^{\bar{y}_f} = \tau(k)$ as well.

Case 2. Assume $i = j$ and $y_1 \dots y_t \in R_2$. We now have

$$\begin{aligned}
 \tau(k) &= \tau(i)^{y_1 \dots y_{f-1}} \\
 \tau(k)^{y_f} &= \tau(i)^{y_1 \dots y_f} && \text{Remark 2.6} \\
 &= \tau(i)^{y_1 \dots y_f y_{f+1} \dots y_t \bar{y}_t \dots \bar{y}_{f+1}} && \text{Prop. 2.5 (1)} \\
 &= \tau(i)^{\bar{y}_t \dots \bar{y}_{f+1}} && \text{Prop. 2.5 (3)} \\
 &= \tau(\ell).
 \end{aligned}$$

As in Case 1, this implies $\tau(\ell)^{\bar{y}_f} = \tau(k)$ as well. \square

Before giving the COINCIDENCE procedure we describe three additional routines MERGE, REP, and UPDATE which will all be used by COINCIDENCE. The procedure REP(i) finds the least representative of i and the related procedure UPDATE(i) changes \mathcal{E} so that $\rho(j) = \text{REP}(i)$ for all $j \in \text{orb}(i)$. Notice that $\rho(i) \leq i$ is required for the procedure REP to find the least representative.

Algorithm 6 Finding the least representative of i

```

1: procedure REP( $i$ )
2:   Input:  $\mathcal{E}$ ,  $i \in \{1, 2, \dots, \omega\}$ 
3:    $j := i$ 
4:   while  $\rho(j) < j$  do
5:      $j := \rho(j)$ 
6:   end while
7:   return  $j$ 
8: end procedure

```

Algorithm 7 Setting $\rho(j) = \text{REP}(i)$ for all $j \in \text{orb}(i)$

```

1: procedure UPDATE( $\sim\mathcal{T}, i$ )
2:   Input:  $\mathcal{E}$ ,  $i \in \{1, 2, \dots, \omega\}$ 
3:    $\epsilon = \text{REP}(i)$ ;  $n := i$ ;  $m := \rho(n)$ 
4:   while  $m < n$  do
5:      $\rho(n) := \epsilon$ ;  $n := m$ ;  $m := \rho(n)$ 
6:   end while
7: end procedure

```

The COINCIDENCE procedure is called when scanning forward and backward produce two distinct values k and ℓ in the same location of the helper table. The procedure changes ρ of the larger of the two values, replaces all occurrences of the larger value with its new smallest representative, and merges information from the larger value's row into the row for its smallest representative. Sometimes the merging of rows will introduce new coincidences. Hence, our procedure must produce a queue of coincidences that will be resolved in order. The MERGE procedure in Algorithm 8 adds to the queue of coincidences and changes values of ρ to record which elements are to be killed.

Algorithm 8 Adding a coincidence to the merge queue

```

1: procedure MERGE( $\sim Q, \sim \rho, m, n$ )
2:   Input:  $\rho$ ,  $Q$  a queue of coincidences,  $m \equiv n$  a coincidence
3:    $\mu := \text{REP}(m)$ ;  $\nu := \text{REP}(n)$ 
4:   if  $\mu \neq \nu$  then
5:     append  $\max(\mu, \nu)$  to  $Q$ 
6:      $\rho(\max(\mu, \nu)) := \min(\mu, \nu)$ 
7:   end if
8: end procedure

```

Notice that only live elements are added to the queue but then are immediately killed. This means that the queue is always a distinct set of dead elements. Notice also that $\text{REP}(m) = \text{REP}(n)$ after a call to $\text{MERGE}(m, n)$. We are now prepared to define COINCIDENCE.

Algorithm 9 Resolving a coincidence $m \equiv n$

```

1: procedure COINCIDENCE( $\sim\mathcal{T}, m, n$ )
2:   Input:  $\mathcal{T}, m, n \in \Omega$ ,  $m \equiv n$  a coincidence
3:    $Q := \phi$ 
4:   MERGE( $\sim Q, \sim \rho, m, n$ )                                 $\triangleright$  queue coincidence  $m \equiv n$ 
5:    $q := 1$ 
6:   while  $q \leq \text{length}(Q)$  do
7:      $d := Q(q); q := q + 1$                                       $\triangleright$  take  $q^{\text{th}}$  element off  $Q$ 
8:     for  $x \in S \cup \bar{S}$  do
9:       if  $d^x = e$  then
10:        undefine  $d^x$  and  $e^{\bar{x}}$                                  $\triangleright$  remove inverse pair
11:         $\delta := \text{REP}(d); \text{UPDATE}(d)$ 
12:         $\epsilon := \text{REP}(e); \text{UPDATE}(e)$ 
13:        if  $\delta^x$  is defined then
14:          MERGE( $\sim Q, \sim \rho, \epsilon, \delta^x$ )                   $\triangleright$  queue new coincidence
15:        else if  $\epsilon^{\bar{x}}$  is defined then
16:          MERGE( $\sim Q, \sim \rho, \delta, \epsilon^{\bar{x}}$ )             $\triangleright$  queue new coincidence
17:        else
18:           $\delta^x := \epsilon; \epsilon^{\bar{x}} := \delta$                  $\triangleright$  add inverse pair
19:        end if
20:      end if
21:    end for
22:  end while
23: end procedure

```

The COINCIDENCE procedure involves an indefinite loop since the length of Q can increase. However, since Q is a finite list of distinct elements from $\{1, 2, \dots, \omega\}$ and since no process in COINCIDENCE changes the number of rows ω of \mathcal{E} , the loop will terminate. We need the following lemma to prove that COINCIDENCE preserves Properties 1–5.

Lemma 3.4. *If $i^y = j$ where $i, j \in \Omega$ and $y \in S \cup \bar{S}$ before a call to COINCIDENCE, then $\text{REP}(i)^y = \text{REP}(j)$ after the call.*

Proof. Notice that Property 2 remains true after a call to COINCIDENCE because entries in \mathcal{E} are only removed or added in inverse pairs by the procedure. The COINCIDENCE routine incrementally builds a queue of distinct elements from $\{1, 2, \dots, \omega\}$ that are all dead and have been removed from \mathcal{E} by the time the procedure has completed. Let Q' denote the final queue created by COINCIDENCE. The initial call to MERGE in line 4 initializes the queue by adding either m or n to it.

Assume $i^y = j$ before a call to COINCIDENCE. If $i, j \notin Q'$, then $i^y = j$ is not removed from \mathcal{E} and after the call we have $i = \text{REP}(i)$ and $j = \text{REP}(j)$. Therefore, $\text{REP}(i)^y = \text{REP}(j)$ after the call. So assume then that i or j is in Q' . We will show that after executing lines 9–20, there exists p and q such that $p^y = q$, $q^{\bar{y}} = p$, $\text{REP}(p) = \text{REP}(i)$, and $\text{REP}(q) = \text{REP}(j)$.

Assume first that $i^y = j$, $i \in Q'$, and if $j \in Q'$, then j appears after i in the queue. We leave the other case to the reader. Since j does not appear before i in Q' , there is a point in the execution of the procedure where we reach line 9 with $d = i$, $x = y$, and $e = j$. Starting at line 10, first $i^y = j$ and $j^{\bar{y}} = i$ are removed

from the table and then $\delta = \text{REP}(i)$ and $\epsilon = \text{REP}(j)$ are defined. There are three cases to consider.

- (1) If $\delta^y = f$, then $f^{\bar{y}} = \delta$ (since Property 2 is satisfied) and a call to $\text{MERGE}(\epsilon, f)$ is made. After this call we have $\text{REP}(f) = \epsilon = \text{REP}(j)$ and $\text{REP}(i) = \delta$. If we now define $p = \delta$ and $q = f$, then $p^y = q$, $q^{\bar{y}} = p$, $\text{REP}(p) = \text{REP}(i)$, and $\text{REP}(q) = \text{REP}(j)$.
- (2) If δ^y is undefined but $\epsilon^{\bar{y}} = f$, then $f^y = \epsilon$ and a call to $\text{MERGE}(\delta, f)$ is made. Similar to above, we have $\text{REP}(f) = \delta = \text{REP}(i)$ and $\epsilon = \text{REP}(j)$ after the merge. In this case, define $p = f$ and $q = \epsilon$ and the result is true.
- (3) If δ^y and $\epsilon^{\bar{y}}$ are both undefined, then we add the entries $\delta^y = \epsilon$ and $\epsilon^{\bar{y}} = \delta$ to \mathcal{E} . Since no values of ρ are changed in this case, we still have that $\text{REP}(i) = \delta$ and $\text{REP}(j) = \epsilon$. In this case, define $p = \delta$ and $q = \epsilon$ and the result is true.

Notice that in every case, neither p nor q can appear before i in Q' .

We are now prepared to prove that if $i^y = j$ before a call to COINCIDENCE , then $\text{REP}(i)^y = \text{REP}(j)$ after the call. Set $i_0 = i$ and $j_0 = j$. If $i_0, j_0 \notin Q'$, then we are done. Otherwise, as shown above, there exists i_1, j_1 such that $i_1^y = j_1$, $\text{REP}(i_0) = \text{REP}(i_1)$, and $\text{REP}(j_0) = \text{REP}(j_1)$. If $i_1, j_1 \notin Q'$, then we are done. Otherwise, note that the first occurrence of either i_0 or j_0 in Q' must precede the first occurrence of i_1 or j_1 by our remark above. Since Q' is finite, this implies that the process must terminate with a last equation $i_\ell^y = j_\ell$ where $i_\ell, j_\ell \notin Q'$. Therefore,

$$\text{REP}(i)^y = \text{REP}(i_\ell)^y = i_\ell^y = j_\ell = \text{REP}(j_\ell) = \text{REP}(j).$$

□

We are now prepared to prove that COINCIDENCE preserves Properties 1–5.

Proposition 3.5. *Properties 1–5 are preserved by each call to COINCIDENCE .*

Proof. Notice that $1 \in \Omega$ after a call to COINCIDENCE because MERGE will never change $\rho(1)$. Furthermore, none of the procedures alter τ so Property 1 remains true. As already seen in the proof of Lemma 3.4, Property 2 remains true after a call to COINCIDENCE .

It is convenient to prove Properties 3 and 5 together. Assume both properties are true before a call to COINCIDENCE . We first consider the initial call to MERGE on line 4. The MERGE procedure does not change values of A nor τ and so Property 3 is still true after line 4. On the other hand, MERGE does change values of ρ and so we must show Property 5 remains true after line 4. There are two cases to consider depending on whether COINCIDENCE was called by SCAN when considering a primary or a secondary relation. First consider $\text{SCAN}(i, y_1 \dots y_t, j)$ with $1 \leq i, j \leq g$ and $x_i^{y_1 \dots y_t} = x_j \in R$ a primary relation. A coincidence occurs when $y_1 \dots y_{t-1} = k$ is defined, $j^{\bar{y}_t \dots \bar{y}_f} = \ell$ is defined, $k \neq \ell$, and $k, \ell \in \Omega$. Hence, $\text{REP}(k) = k$ and $\text{REP}(\ell) = \ell$ before the call to COINCIDENCE . Since Property 3 is true before the call, we have that $\tau(k) = x_i^{y_1 \dots y_{t-1}}$ and $\tau(\ell) = x_j^{\bar{y}_t \dots \bar{y}_f}$. Therefore,

$$\begin{aligned} \tau(k) &= x_i^{y_1 \dots y_{t-1}} \\ &= x_i^{y_1 \dots y_{t-1} y_f \dots y_t \bar{y}_t \dots \bar{y}_f} && \text{Prop. 2.5 (1)} \\ &= x_j^{\bar{y}_t \dots \bar{y}_f} && \text{Prop. 2.5 (2)} \\ &= \tau(\ell). \end{aligned}$$

Assume that $k > \ell$ and so after the call to $\text{MERGE}(k, \ell)$ in line 4 we have $\text{REP}(k) = \text{REP}(\ell) = \ell$. Therefore, $\tau(\ell) = \tau(\text{REP}(\ell))$ and $\tau(k) = \tau(\ell) = \tau(\text{REP}(k))$ so Property 5 remains true after line 4. The case where $k < \ell$ is similar as is the case when SCAN is applied to a secondary relation.

We next show that the two properties are preserved by inducting on the number of times lines 4 and 13 are executed. The previous argument for the call to MERGE in line 4 establishes the base case. So assume that both properties are true and we arrive at line 13 with $d \in Q$, $x \in S \cup \bar{S}$, $d^x = e$, $\delta = \text{REP}(d)$, and $\epsilon = \text{REP}(e)$. Thus, at this point, $\tau(d)^x = \tau(e)$, $\tau(d) = \tau(\delta)$, and $\tau(e) = \tau(\epsilon)$ by our inductive hypothesis. There are three cases to consider: we may call MERGE on lines 14 or 16, or add two entries to the table on line 18. In no case are values of τ changed, but MERGE may change values of ρ . Thus, Property 5 will remain true if line 18 is executed and we must still show that it remains true if line 14 or 16 is executed. Similarly, Property 3 will remain true if line 14 or 16 is executed and we must still show that it remains true if line 18 is executed.

Suppose $\delta^x = f$ and we call $\text{MERGE}(\epsilon, f)$ in line 14. So, by Property 3, $\tau(\delta)^x = \tau(f)$. Suppose now that ℓ is arbitrary. We want to show that $\tau(\ell) = \tau(\text{REP}(\ell))$ after the call to $\text{MERGE}(\epsilon, f)$. Assume that $\epsilon > \text{REP}(f) = \phi$ in which case MERGE will set $\rho(\epsilon) = \phi$. If before the call to MERGE, $\text{REP}(\ell) \neq \epsilon$, then $\text{REP}(\ell)$ will be unchanged and after the call we will still have $\tau(\ell) = \tau(\text{REP}(\ell))$. However, if $\text{REP}(\ell) = \epsilon$ before the call, then $\tau(\ell) = \tau(\epsilon)$ and, after the call, we will have $\text{REP}(\ell) = \phi$. Using all of the above, we have

$$\tau(\ell) = \tau(\epsilon) = \tau(e) = \tau(d)^x = \tau(\delta)^x = \tau(f) = \tau(\phi) = \tau(\text{REP}(\ell)).$$

The case where $\epsilon < \phi$ is similar. If δ^x is undefined and $\epsilon^{\bar{x}}$ is defined, then the argument is similar.

Consider now the third possibility where δ^x and $\epsilon^{\bar{x}}$ are both undefined. In this case, two entries $\delta^x = \epsilon$ and $\epsilon^{\bar{x}} = \delta$ are added to \mathcal{E} by line 18. By the inductive hypotheses, we have

$$\tau(\delta)^x = \tau(d)^x = \tau(e) = \tau(\epsilon).$$

Therefore, Property 3 remains true after executing line 18.

Finally, consider Property 4. If $j \in \Omega$ after the call to COINCIDENCE, then before the call, $j \in \Omega$ and there exists $i \in \Omega \cap \{1, 2, \dots, g\}$ and $w \in F(S)$ such that $j = i^w$. Because $j \in \Omega$ after the call we have $j = \text{REP}(j)$. Moreover, from Lemma 3.4, $j = \text{REP}(j) = \text{REP}(i)^w$ after the call. Because $\text{REP}(i) \leq i$, this establishes Property 4. \square

Combining the results in this section we have the following theorem.

Theorem 3.6. *If $\text{ENUMERATE}(S, R, M)$ completes, then \mathcal{E} is complete and satisfies Properties 1–5.*

Note that even if $\text{ENUMERATE}(S, R, M)$ returns a run limit exceeded statement, then the table produced up to that point still satisfies Properties 1–5. The table may even be complete, however, the secondary relations have not been scanned for all $i \in \Omega$.

4. COMPLETE TABLES AND THE RACK ISOMORPHISM

In this section we establish our main results regarding the relationship between \mathcal{E} and \mathcal{R} when ENUMERATE completes. We begin with a useful lemma.

Lemma 4.1. *If $\text{ENUMERATE}(S, R, M)$ completes, then for all $i, j \in \{1, 2, \dots, \omega\}$ we have $\text{REP}(i) = \text{REP}(j)$ if and only if $\tau(i) = \tau(j)$ in \mathcal{R} .*

Proof. By Theorem 3.6 we know that \mathcal{E} satisfies Properties 1–5. Therefore, if $\text{REP}(i) = \text{REP}(j)$, then by Property 5 we have

$$\tau(i) = \tau(\text{REP}(i)) = \tau(\text{REP}(j)) = \tau(j).$$

Conversely, assume that $\tau(i) = \tau(j)$. Since \mathcal{E} satisfies Properties 1–5 and $\text{REP}(i), \text{REP}(j) \in \Omega$, there exists $a, b \in \Omega \cap \{1, 2, \dots, g\}$ and $\alpha, \beta \in F(S)$ such that $a^\alpha = \text{REP}(i)$, $b^\beta = \text{REP}(j)$, $\tau(\text{REP}(i)) = x_a^\alpha$, and $\tau(\text{REP}(j)) = x_b^\beta$. We now have

$$x_a^\alpha = \tau(\text{REP}(i)) = \tau(i) = \tau(j) = \tau(\text{REP}(j)) = x_b^\beta.$$

Therefore, there is a finite sequence of substitution moves that take x_a^α to x_b^β . We will show that for each substitution move, if x_e^u is taken to x_f^v , then $\text{REP}(e)^u = \text{REP}(f)^v$ in \mathcal{E} . Therefore, after the finite sequence of moves that takes x_a^α to x_b^β , we have

$$\text{REP}(i) = a^\alpha = \text{REP}(a)^\alpha = \text{REP}(b)^\beta = b^\beta = \text{REP}(j).$$

Move (1): Assume move (1) takes x_e^{uw} to $x_e^{uv\bar{v}w}$. Since \mathcal{E} is complete and satisfies Property 2 we have that $\ell^{v\bar{v}} = \ell$ for all $\ell \in \Omega$. Therefore, $\text{REP}(e)^{uw} = \text{REP}(e)^{uv\bar{v}w}$.

Move (2): Assume move (2) takes x_e^{uw} to x_f^w where $x_e^u = x_f$ is a primary relation. Then $\text{SCAN}(\text{REP}(e), u, \text{REP}(f))$ was called in the ENUMERATE procedure. Therefore, $\text{Rep}(e)^u = \text{Rep}(f)$ in \mathcal{E} and so $\text{Rep}(e)^{uw} = \text{Rep}(f)^w$.

Move (3): Assume move (3) takes x_e^{vw} to $x_e^{v\bar{u}x_cu\bar{x}_d w}$ where $x_c^u = x_d$ is a primary relation. Then, $\text{SCAN}(\text{Rep}(e)^v, \bar{u}x_cu\bar{x}_d, \text{Rep}(e)^v)$ was called in the ENUMERATE procedure because $\text{Rep}(e)^v \in \Omega$. Therefore, $\text{Rep}(e)^{v\bar{u}x_cu\bar{x}_d} = \text{Rep}(e)^v$ in \mathcal{E} and so $\text{Rep}(e)^{v\bar{u}x_cu\bar{x}_d w} = \text{Rep}(e)^{vw}$. The other case of substitution move (3) is similar. \square

Notice that if $\text{ENUMERATE}(S, R, M)$ completes, then for all $j \in \Omega$ there exists $k \in \Omega \cap \{1, 2, \dots, g\}$ and $w \in F(S)$ such that $j = k^w$. In this case we may define two operations on Ω by

$$(1) \quad i^j = i^{\bar{w}x_k w} \quad \text{and} \quad i^{\bar{j}} = i^{\bar{w}\bar{x}_k w}.$$

Theorem 4.2. *If $\text{ENUMERATE}(S, R, M)$ completes, then Ω with operations given by (1) is a rack and $\tau : \Omega \rightarrow \mathcal{R}$ is a rack isomorphism.*

Proof. We first show the operations are well-defined. Assume $j = k^w = \ell^v$ with $k, \ell \in \Omega \cap \{1, 2, \dots, g\}$ and $w, v \in F(S)$. Since \mathcal{E} is complete, k^w and ℓ^v are both in Ω and hence, by Lemma 4.1, $\tau(k^w) = \tau(\ell^v)$. Property 3 now implies that $\tau(k)^w = \tau(\ell)^v$ and by Property 1, $x_k^w = x_\ell^v$. Hence, the rack axioms tell us that $\tau(i)^{\bar{w}x_k w} = \tau(i)^{\bar{v}x_\ell v}$ and using Property 3 again, $\tau(i^{\bar{w}x_k w}) = \tau(i^{\bar{v}x_\ell v})$. Now by Lemma 4.1, $i^{\bar{w}x_k w} = i^{\bar{v}x_\ell v}$. Therefore i^j is well-defined. The proof for $i^{\bar{j}}$ is similar.

Suppose that $i, j \in \Omega$ and $j = k^w$. We have

$$(i^j)^{\bar{j}} = (i^{\bar{w}x_k w})^{\bar{j}} = i^{\bar{w}x_k w\bar{w}\bar{x}_k w} = i.$$

Similarly, $(i^{\bar{j}})^j = i$. Thus, the first rack axiom holds.

Let $\ell = m^v$. Then

$$(i^j)^\ell = (i^{\bar{w}x_k w})^\ell = i^{\bar{w}x_k w\bar{v}x_m v}$$

and

$$(i^\ell)^{(j^\ell)} = (i^{\bar{v}x_m v})^{(j^{\bar{v}x_m v})} = (i^{\bar{v}x_m v})^{(k^w\bar{v}x_m v)} = i^{\bar{v}x_m v\bar{v}\bar{x}_m v\bar{w}x_k w\bar{v}x_m v} = i^{\bar{w}x_k w\bar{v}x_m v}.$$

Therefore, the second rack axiom is satisfied.

The function $\tau : \Omega \rightarrow \mathcal{R}$ is injective by Lemma 4.1. Now suppose $x_i^w \in \mathcal{R}$ where $1 \leq i \leq g$. Then $\text{REP}(i) \in \Omega$ and Properties 1–5 imply

$$\tau(\text{REP}(i)^w) = \tau(\text{REP}(i))^w = \tau(i)^w = x_i^w.$$

Therefore, τ is also surjective.

Finally, assume $i, j \in \Omega$ with $j = k^w$. Now

$$\tau(i^j) = \tau(i^{\bar{w}x_k w}) = \tau(i)^{\bar{w}x_k w}.$$

On the other hand

$$\tau(i)^{\tau(j)} = \tau(i)^{\tau(k^w)} = \tau(i)^{(\tau(k))^w} = \tau(i)^{(x_k^w)} = \tau(i)^{\bar{w}x_k w}.$$

Thus, $\tau(i^j) = \tau(i)^{\tau(j)}$. Hence, $\tau(i) = \tau(i^{\bar{j}}j) = \tau(i^{\bar{j}})^{\tau(j)}$. It now follows that $\tau(i^{\bar{j}}) = \tau(i)^{\tau(j)}$. Therefore, τ is a rack isomorphism. \square

An important step in the ENUMERATE procedure is the DEFINE command in line 18 which represents Ward's modification of the Todd-Coxeter process in the rack setting. This line requires that after scanning and filling all secondary relations for row $i \in \Omega$ we make additional definitions, if necessary, so that i^y is defined for all $y \in S \cup \bar{S}$. We do this before moving to the next live row. While this step can increase the size of Ω , it has the benefit of producing a table that is filled in through row i after completing all secondary relation scans for row i . This is important in the proof of the following theorem.

Theorem 4.3. *If $\mathcal{R} = \langle S \mid R \rangle$ is a finite rack, then $\text{ENUMERATE}(S, R, M)$ completes for some M .*

Proof. Towards contradiction, assume \mathcal{R} is finite and that $\text{ENUMERATE}(S, R, M)$ returns a run limit exceeded statement for all $M \geq 1$. For a fixed M , let Ω_M be the live elements at the completion of $\text{ENUMERATE}(S, R, M)$ and define $\bar{\Omega} = \cap_{M \geq 1} \Omega_M$. Thus, $\bar{\Omega}$ is the set of elements that are not killed in any call of COINCIDENCE. By Property 1, we have $1 \in \bar{\Omega}$, so this set is nonempty. Now if $i \in \bar{\Omega}$ and $y \in S \cup \bar{S}$, then for all $M \geq i$ we have that line 18 of $\text{ENUMERATE}(S, R, M)$ guarantees that i^y is defined. Notice that, as we increase M , the values of i^y are nonincreasing since COINCIDENCE replaces dead values with their least representative. The values of i^y are bounded below by 1, therefore, at some point i^y becomes stable. Since $S \cup \bar{S}$ is finite, this implies that given $i \in \bar{\Omega}$ there is an $M_i \geq i$ such that i^y is defined and stable for all $M \geq M_i$ and for all $y \in S \cup \bar{S}$. Notice also that the stable value of i^y is in $\bar{\Omega}$ for all $y \in S \cup \bar{S}$.

If $\bar{\Omega}$ were finite, then the set $\{M_i \mid i \in \bar{\Omega}\}$ would also be finite. In this case, we could choose $N \geq \max\{M_i \mid i \in \bar{\Omega}\}$ and $\text{ENUMERATE}(S, R, N)$ would create an enumeration table in which i^y is defined and $i^y \in \bar{\Omega}$ for all $i \in \bar{\Omega}$ and $y \in S \cup \bar{S}$. Hence, for all $w \in F(S)$, we would have $i^w \in \bar{\Omega}$ as well. Increase N , if necessary, so that $\Omega_N \cap \{1, 2, \dots, g\} = \bar{\Omega} \cap \{1, 2, \dots, g\}$. Now by Property 4, for any $n \in \Omega_N$, there exists an $i \in \bar{\Omega} \cap \{1, 2, \dots, g\}$ and $w \in F(S)$ such that $i^w = n$. However, $i \in \bar{\Omega}$ implies $i^w = n \in \bar{\Omega}$ and, hence, $\bar{\Omega} = \Omega_N$. Since $N \geq M_i \geq i$ for all $i \in \bar{\Omega} = \Omega_N$, this implies that $\text{ENUMERATE}(S, R, N)$ completes. Hence we have a contradiction. Therefore, $\bar{\Omega}$ must be infinite.

Now consider the infinite enumeration table \mathcal{T}_∞ whose (infinitely many) rows are the elements of $\bar{\Omega}$ and whose entries are the stable values of i^y for $i \in \bar{\Omega}$. This table is complete and satisfies Properties 1–5. Therefore, by the argument in

Theorem 4.2, we have that $\tau : \overline{\Omega} \rightarrow \mathcal{R}$ is a rack isomorphism. This contradicts that \mathcal{R} is finite. \square

The following is an immediate corollary of Theorems 4.2 and 4.3.

Corollary 4.4. *A finitely presented rack $\mathcal{R} = \langle S \mid R \rangle$ is finite if and only if there is an M such that $\text{ENUMERATE}(S, R, M)$ completes.*

5. MODIFICATIONS TO ENUMERATE

Recall that the `DEFINE` command in line 18 of the `ENUMERATE` procedure represents Ward's modification to the Todd-Coxeter procedure in the rack setting. Ward's modification was motivated by examples, given in [11], where the original Todd-Coxeter process failed to complete even though the subgroup index was finite. A similar example exists in the rack setting. Consider the rack \mathcal{R} with presentation

$$(2) \quad \langle a, b \mid a^a = a, b^b = b, a^{babab} = a, b^{babab} = b \rangle.$$

Then $\text{ENUMERATE}(S, R, 11)$ completes with \mathcal{E} , shown below.

\mathcal{E}	a	b	\bar{a}	\bar{b}	τ	p
1	1	1	1	1	a	1
2	6	2	6	2	b	2
6	2	6	2	6	b^a	6

From the table it is clear that \mathcal{R} is an involutory quandle of order 3. However, if we omit the command in line 18 of `ENUMERATE`, then the process never completes.

Proposition 5.1. *If $\text{W}(S, R, M)$ is the rack enumeration procedure with line 18 omitted, then there is no M for which $\text{W}(S, R, M)$ completes for the finite rack presented by (2).*

Strong evidence for the veracity of the proposition can be obtained by coding the `ENUMERATE` procedure with line 18 omitted and running it for presentation (2) with large values of M . A formal proof of the remark can be given by using induction to prove that there is a sequence $4 = n_0 < n_1 < n_2 < n_3 < \dots$ with the property that, for all $i \geq 1$, $\text{W}(S, R, n_i - 1)$ produces a run limit exceeded statement and an incomplete table which contains the following lines (here, a 0 in \mathcal{E} represents an undefined entry).

\mathcal{T}	a	b	\bar{a}	\bar{b}	ρ
n_{i-1}	n_i	*	*	0	4
n_i	0	$n_i + 1$	n_{i-1}	0	n_i
n_{i+1}	$n_i + 2$	0	$n_i + 3$	n_i	$n_i + 1$
n_{i+2}	0	$n_i + 3$	$n_i + 1$	0	$n_i + 2$
n_{i+3}	$n_i + 1$	0	0	$n_i + 2$	$n_i + 3$

The inductive step requires a careful analysis of the helper tables for the scans of the secondary relations for rows 1 through $n_{i+1} - 1$ and how they affect the entries of \mathcal{E} . We leave the details for the interested reader.

The inclusion of line 18 is a simple way to avoid the problem in Proposition 5.1, however, there may be other ways to alter ENUMERATE to achieve this. For example, our INIT routine does not attempt to produce the most efficient set of secondary relations. Notice that if $x^w = x$ for all x , then $x^{\bar{w}} = x$ for all x as well. For this reason it is unnecessary to have both w and \bar{w} in the set R_2 of secondary relations. Similarly, if $x^{uw} = x$ is a secondary relation and given any $y \in \mathcal{R}$, if we let $x = y^{\bar{u}}$, then the secondary relation gives $y^{\bar{u}uv} = y^u$. Hence, $y^{vu} = y$ for all $y \in \mathcal{R}$. Therefore, given any secondary relation $x^w = x$, we are free to cyclically permute the letters in w to obtain an equivalent secondary relation. This allows us to record any secondary relation $x^w = x$ with the unique word w' which is *minimal* amongst all words obtained from w and \bar{w} by cyclic permutation and reduction. Here minimal means of shortest length and, among words of the same length, lexicographically smallest where the order on $S \cup \bar{S}$ is $x_1 < x_2 < \dots < x_g < \bar{x}_g < \dots < \bar{x}_1$.

Consider once more the rack \mathcal{R} defined by (2). With INIT defined by Algorithm 2, the set of secondary relation words is $R_2 = \{bab\bar{a}babab\bar{a}, b\bar{a}\bar{b}abab\bar{b}\}$. On the other hand, if instead we consider the minimal representatives of these words, then our secondary relations would be $R'_2 = \{abab\bar{a}b\bar{a}\bar{b}, abab\bar{a}\bar{b}\}$. Running a modification of ENUMERATE with line 18 omitted and R_2 replaced by R'_2 , the process completes. We do not know if this is true in general, that is, if the secondary relations are chosen in this way, then is line 18 still necessary?

As mentioned in the introduction, the Todd-Coxeter process was designed to find the index of a finitely generated subgroup of a group. That is, it is designed to enumerate cosets. The order and Cayley graph for the group can be found by enumerating the cosets of the trivial subgroup. It is natural to ask if the ENUMERATE process can be modified to enumerate something more general than the elements of the rack. The natural analogy is to consider a finitely generated subrack $\Sigma \subseteq \mathcal{R} = \langle S \mid R \rangle$. If $x \in \mathcal{R}$, then define the **rack coset** Σ^x to be the set of elements $\{\sigma^x \mid \sigma \in \Sigma\}$. Unfortunately, the collection of all rack cosets of a given subrack does not, in general, partition the rack. We consider three interesting examples.

First, consider the fundamental 4-quandle of the righthand trefoil knot given by the presentation

$$\mathcal{R}_1 = \langle a, b \mid a^a = a, b^b = b, a^{bbbb} = a, b^{aaaa} = b, a^{ba} = b, b^{ab} = a \rangle.$$

From the ENUMERATE process we find that $\mathcal{R}_1 = \{a, b, a^b, b^a, a^{bb}, b^{aa}\}$. Moreover, it is not hard to show that $\Sigma = \langle a, a^{bb} \rangle = \{a, a^{bb}\}$ is a subrack (in fact, a subquandle). By direct calculation, this subrack has three distinct cosets Σ , Σ^b , and Σ^{ba} which partition \mathcal{R}_1 . The ENUMERATE process can be modified to enumerate these cosets and determine the action of \mathcal{R}_1 on the cosets. Namely, initialize the process with 1 representing the coset Σ . Since Σ is a subrack, it is fixed by the action of both generators a and a^{bb} . So the modified process first scans these two subrack generator relations: $1^a = 1$ and $1^{\bar{b}bab\bar{b}} = 1$. Next the modified process scans all secondary relations (from the presentation of \mathcal{R}_1) and an additional secondary relation $i^{\bar{b}bab\bar{a}} = i$, since the action by different generators of Σ should be the same, for all live i . The modified process completes and enumerates the three distinct cosets given above.

As a second example, consider the involutory quandle of the $(2, 4)$ -torus link which has a presentation

$$\mathcal{R}_2 = \langle a, b \mid a^a = a, b^b = b, a^{bb} = a, b^{aa} = b, a^{bab} = a, b^{aba} = b \rangle.$$

The ENUMERATE process determines that $\mathcal{R}_2 = \{a, b, a^b, b^a\}$. This rack contains the subrack $\Sigma = \langle a, a^b \rangle = \{a, a^b\}$. Notice that $\Sigma = \Sigma^a = \Sigma^b = \Sigma^{a^b} = \Sigma^{b^a}$ and so the cosets of the subrack do not partition \mathcal{R}_2 . On the other hand, consider the rack with presentation

$$\mathcal{R}_3 = \langle a, b, c \mid a^a = a, b^b = b, c^c = c, a^b = a, a^c = a, b^a = b, c^a = c, b^{cc} = c, b^{cbb} = c \rangle$$

which has order 6. The subrack $\Sigma = \langle a, b \rangle = \{a, b\}$ has five distinct cosets all of which contain a and whose union is the entire rack. It is not immediately clear how the enumeration process can be modified in these last two examples in order to enumerate the distinct cosets.

6. AN APPLICATION TO KNOT THEORY

We close with a sample calculation related to knot theory. Associated to every link is its fundamental quandle, which, of course, is a rack. However, the quandle of a knot or link is almost always infinite. If we pass to the quotient 2-quandle, then there are many knots and links for which this is finite. A complete list of links with finite n -quandles for some n is given in [5]. One such link is shown in Figure 1.

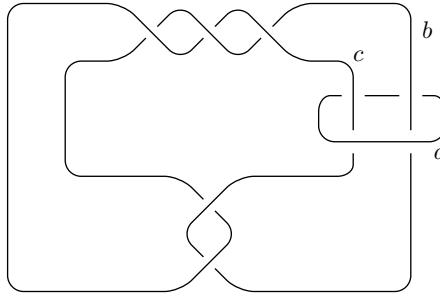


FIGURE 1. A link with finite 2-quandle.

A presentation for the 2-quandle of the link can be obtained from the diagram by labeling each arc of the diagram with a generator and then recording one relation at every crossing as indicated in Figure 2. In addition to these relations, we must also include the relations $x^x = x$ for every generator and $x^{yy} = x$ for every pair of distinct generators x and y . See [5] for more information on presentations of 2-quandles of links. If we use one generator for each arc, we will create a presentation with redundant generators. Instead, it is always possible to label some subset of the arcs with generators and then use the relations at each crossing to derive the labels on all of the other arcs. Arcs for which labels can be so derived in two different ways then give rise to the necessary relations.

If we label the three arcs shown in Figure 1 with the generators a , b , c , and follow the above procedure, we obtain the presentation

$$\begin{aligned} \mathcal{R} = \langle a, b, c, & \mid a^a = a, b^b = b, c^c = c, \\ & a^{bb} = a, a^{cc} = a, b^{aa} = b, b^{cc} = b, c^{aa} = c, c^{bb} = c, \\ & a^{bc} = a, b^{acbebcba} = c, b^{cbcacacb} = c \rangle. \end{aligned}$$

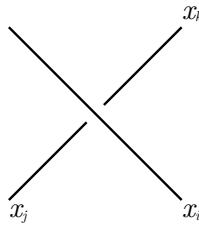


FIGURE 2. This crossing gives the relation $x_j^{x_i} = x_k$.

Applying ENUMERATE to this presentation yields a finite 2-quandle with two algebraic components corresponding to the two components of the link. One algebraic component has four elements including the generator a and the other has twenty elements including the generators b and c . The Cayley graph of the 2-quandle can be immediately derived from the enumeration table and is shown in Figure 3. The generators a, b , and c , correspond to the solid, dashed, and dotted edges, respectively.

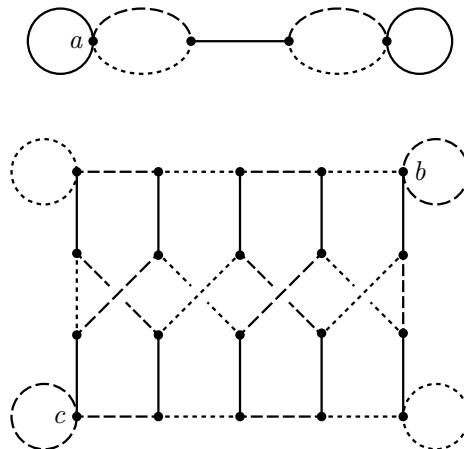


FIGURE 3. The 2-quandle of the link in Figure 1.

7. IMPLEMENTATION

Since its implementation on a computer, there have been multiple modifications made to the Todd-Coxeter process that decrease the run-time or memory usage. In [3], Holt, Eick, and O'Brien characterize the performance of a coset enumeration process in terms of the maximum number of live elements at any stage of the process. That is, the maximum value of $|\Omega|$ at any point. They also remark that the total number of cosets defined would also be a reasonable measure. Holt, Eick, and O'Brien declare a procedure to perform well if $\max\{|\Omega|\}$ is roughly less than 125% of the index $[G : H]$.

TABLE 1. Performance of ENUMERATE for a sample of Montesinos link quandles.

p	q	e	t	\mathcal{O}	L	E	L/\mathcal{O}	E/\mathcal{O}
2	23	2	0.11	1008	4646	8109	4.6	8.0
53	61	2	0.20	992	9832	22148	9.9	22.3
2	49	-1	4.61	10000	117876	615021	11.8	61.5
2	11	5	0.24	1008	13893	30482	13.8	30.2
2	61	5	25.84	30008	559137	2483138	18.6	82.7
4	41	4	4.91	9996	198559	593150	19.9	59.3
31	39	5	7.56	10000	367832	1039894	36.8	104.0
4	49	-3	18.73	20000	838911	2312936	41.9	115.6
5	9	-4	0.46	1000	47906	64245	47.9	64.2
19	45	-1	10.11	10028	500924	1132344	50.0	112.9
27	53	5	28.64	19980	1194349	3942721	59.8	197.3
39	64	-2	87.98	30030	1954031	4726305	65.1	157.4
19	52	5	34.04	20034	1394756	4635357	69.6	231.4
25	64	5	128.65	30030	2321654	8237209	77.3	274.3
31	57	-3	126.44	30044	3027595	7312811	100.8	243.4
12	43	-4	35.56	19976	2069917	4988150	103.6	249.7
16	39	-5	41.27	20000	2507287	5651463	125.4	282.6
17	27	-5	16.27	10024	1334984	2521252	133.2	251.5
31	47	-5	166.36	30048	4338376	9511360	144.4	316.5

We apply this analysis to the enumeration of involutory quandles of a family of links. In [4], it was shown that the order of the involutory quandle of the $(1/2, 1/2, p/q; e)$ -Montesinos link is $2(q+1)|(e-1)q-p|$. A selection of 21, 15, and 16, such quandles with orders near 10000, 20000, and 30000, respectively, were used. Run-times varied from 3.8 to 16.3 seconds for the first group, 18.7 to 41.3 seconds for the second group, and 24.2 to 166.4 seconds for the last group. Not surprisingly, run-times were roughly proportional to the number of elements defined during execution. This number ranged from about 40 to 320 times the order of the quandle. The largest number of live elements during execution was generally between 12 to 145 times the order of the quandle. We coded our implementation of ENUMERATE using *Python* and our program may be downloaded from the CompuTop.org software archive. A small selection of data is included in Table 1. In this table, t is the run-time in seconds, L is the maximum value of $|\Omega|$ at any point in the process, E is the total number of quandle elements defined by the process, and \mathcal{O} is the order of the quandle. We suspect that our procedure could be improved in order to perform well with respect to Holt's measure.

ACKNOWLEDGMENTS

The authors thank the referees for their detailed and helpful comments on the article.

REFERENCES

- [1] M. Elhamdadi and S. Nelson, *Quandles—An Introduction to the Algebra of Knots*, Student Mathematical Library, vol. 74, American Mathematical Society, Providence, RI, 2015. MR3379534
- [2] R. Fenn and C. Rourke, *Racks and Links in Codimension Two*, J. Knot Theory Ramifications **1** (1992), no. 4, 343–406, DOI 10.1142/S0218216592000203. MR1194995
- [3] D. F. Holt, B. Eick, and E. A. O’Brien, *Handbook of Computational Group Theory*, Discrete Mathematics and its Applications (Boca Raton), Chapman & Hall/CRC, Boca Raton, FL, 2005. MR2129747
- [4] J. Hoste and P. D. Shanahan, *Involutory quandles of $(2, 2, r)$ -Montesinos links*, J. Knot Theory Ramifications **26** (2017), no. 3, 1741003, 19, DOI 10.1142/S0218216517410036. MR3627703
- [5] J. Hoste and P. D. Shanahan, *Links with finite n -quandles*, Algebr. Geom. Topol. **17** (2017), no. 5, 2807–2823, DOI 10.2140/agt.2017.17.2807. MR3704243
- [6] D. L. Johnson, *Presentations of Groups*, 2nd ed., London Mathematical Society Student Texts, vol. 15, Cambridge University Press, Cambridge, 1997. MR1472735
- [7] D. Joyce, *A classifying invariant of knots, the knot quandle*, J. Pure Appl. Algebra **23** (1982), no. 1, 37–65, DOI 10.1016/0022-4049(82)90077-9. MR638121
- [8] D. E. Joyce, *An algebraic approach to symmetry with applications to knot theory*, ProQuest LLC, Ann Arbor, MI, 1979. Thesis (Ph.D.)—University of Pennsylvania. MR2628474
- [9] S. V. Matveev, *Distributive groupoids in knot theory* (Russian), Mat. Sb. (N.S.) **119(161)** (1982), no. 1, 78–88, 160. MR672410
- [10] J. A. Todd and H. S. M. Coxeter, *A practical method for enumerating cosets of a finite abstract group*, Proceedings of the Edinburgh Mathematical Society, Series II, 5, pp. 26–34, (1936).
- [11] J. N. Ward, *A note on the Todd-Coxeter algorithm*, Group theory (Proc. Miniconf., Australian Nat. Univ., Canberra, 1975), Springer, Berlin, 1977, pp. 126–129. Lecture Notes in Math., Vol. 573. MR0447411
- [12] S. K. Winker, *Quandles, knot invariants, and the n -fold branched cover*, ProQuest LLC, Ann Arbor, MI, 1984. Thesis (Ph.D.)—University of Illinois at Chicago. MR2634013

DEPARTMENT OF MATHEMATICS, PITZER COLLEGE, 1050 N MILLS AVENUE, CLAREMONT, CALIFORNIA 91711

Email address: jhoste@pitzer.edu

DEPARTMENT OF MATHEMATICS, LOYOLA MARYMOUNT UNIVERSITY, UHALL 2700, LOS ANGELES, CALIFORNIA 90045

Email address: pshanahan@lmu.edu