CORE
Metadata, citation and similar papers at core.ac.uk
Provided by St Andrews Research Repository

arXiv:1510.01868v1 [math.RA] 7 Oct 2015

Computing finite semigroups

J. East, A. Egri-Nagy, J. D. Mitchell, and Y. Péresse

October 8, 2015

Abstract

Using a variant of Schreier's Theorem, and the theory of Green's relations, we show how to reduce the computation of an arbitrary subsemigroup of a finite regular semigroup to that of certain associated subgroups. Examples of semigroups to which these results apply include many important classes: transformation semigroups, partial permutation semigroups and inverse semigroups, partition monoids, matrix semigroups, and subsemigroups of finite regular Rees matrix and 0-matrix semigroups over groups. For any subsemigroup of such a semigroup, it is possible to, among other things, efficiently compute its size and Green's relations, test membership, factorize elements over the generators, find the semigroup generated by the given subsemigroup and any collection of additional elements, calculate the partial order of the $\mathscr{D}$-classes, test regularity, and determine the idempotents. This is achieved by representing the given subsemigroup without exhaustively enumerating its elements. It is also possible to compute the Green's classes of an element of such a subsemigroup without determining the global structure of the semigroup.

Contents

List of Algorithms

List of Figures

1 Introduction

A *semigroup* is a set with an associative binary operation. There are many articles in the literature concerned with the idea of investigating semigroups using a computer; early examples are [5,6,12,18,33]. There are also several examples of software packages specifically for computing with semigroups, such as AUTOMATE [7], Monoid [22], SgpWin [27], Semigroupe [34], and more general computational algebra systems, such as Magma [4], GAP [15], and Sage [42], with some functionality relating to semigroups; see also [8].

Semigroups are commonly represented either by presentations (abstract generators and defining relations) or by a generating set consisting of a specific type of element, such as transformations, matrices, or binary relations. In this paper, we are solely concerned with semigroups defined by generators.

Computing with semigroups defined by generators or with finitely presented semigroups is **hard**; it is shown in [3] that testing membership in a finite commutative transformation semigroup is NP-complete, and it is well-known that determining any 'sensible' property of a finitely presented semigroup is undecidable by the famous results of Post [35] and Markov [26]. However, in spite of the fact that the general case is hard, it is still possible to compute with semigroups efficiently in many particular instances. Perhaps more importantly, it is possible to perform calculations using a computer that it would be impossible (several times over) to do by hand.

Algorithms, and their implementations, for computing semigroups defined by a generating set fall into two classes: those that exhaustively enumerate the elements, and those that do not. Examples of the first type are the algorithms described in [13] and implemented in Semigroupe [34], and those in SgpWin [27]. Exhaustively enumerating and storing the elements of a semigroup quickly becomes impractical. To illustrate, a *transformation* is a function from the set $\{1, \ldots, n\}$ to itself for some $n \in \mathbb{N}$. A semigroup whose elements are transformations and whose operation is composition of functions is called a *transformation semigroup*. For example, if each of the $9^9 = 387420489$ transformations on a 9-element set is stored as a tuple of 9 integers from 1 to 9 and α is the number of bits required to store such an integer, then

$$9^{9+1} \cdot \alpha = 3486784401 \cdot \alpha \text{ bits}$$

are required to store these transformations. In GAP, for example, such an integer requires 16-bits, and so approximately 6 gigabytes of memory would be required in this case. Therefore if we want to exhaustively enumerate a semigroup, then we must be happy to do so with relatively small semigroups. Exhaustive algorithms have the advantage that they are relatively straightforward to apply; if the multiplication of a class of semigroups can be defined to a computer, then these algorithms can be applied. For example, in Semigroupe [34] it is possible to compute with semigroups of transformations, partial transformations, and several types of matrix semigroups including boolean matrices.

Non-exhaustive algorithms are described in [19, 20, 23, 24], and the latter were implemented in the Monoid package [22] for GAP 3 and its later incarnations in GAP 4. In examples where it is not possible to store the elements, these methods can be used to determine structural information about a semigroup, such as its size and Green's relations (see Section 2 for the relevant definitions). In many examples, the non-exhaustive algorithms have better performance than their exhaustive analogues. However, on the down side, the non-exhaustive algorithms described in [20,23,24] only apply to transformation semigroups. The methods in [19] are analogues of the methods in [20] in the context of semigroups of binary relations; but an implementation does not appear to be readily available.

To one degree or another, the articles [3, 19, 20, 23, 24] use variants of Schreier's Theorem [45, Theorem 2.57] and the theory of Green's relations to reduce the computation of the semigroup to that of its Schützenberger groups. It is then possible to use the well-developed and efficient algorithms from Computational Group Theory [39, 41, 45], stemming from the Schreier-Sims Algorithm, to compute with these subgroups. In this paper, we go one step further by giving a computational paradigm for arbitrary subsemigroups of finite regular semigroups. Semigroups to which the paradigm can be efficiently applied include many of the most important classes: transformation semigroups, partial permutation semigroups and inverse semigroups, partition monoids, matrix semigroups, and subsemigroups of finite regular Rees matrix and 0-matrix semigroups. We generalise and improve the central notions in [20,23,24] from transformation semigroups to arbitrary subsemigroups of an arbitrary finite regular semigroup. For such a subsemigroup, it is possible to efficiently compute its size and Green's classes, test membership, factorize elements over the generators, find the semigroup generated by the given subsemigroup and any collection of additional elements, calculate the partial order of the $\mathscr{D}$-classes, test regularity, and determine the idempotents. This is achieved by representing the given subsemigroup without exhaustively enumerating its elements. In particular, our methods can be used to determine properties of semigroups, where it is not possible to store every element of that semigroup. It is also possible to compute the Green's classes of an element of such a subsemigroup without determining the global structure of the semigroup.

Although not described here, it is also possible to use the data structures provided to find the group of units (if it exists), minimal ideal, find a small generating set, and test if a semigroup satisfies various properties such as being simple, completely regular, Clifford and so on. The algorithms described in this paper are implemented in their full generality in the GAP [15] package Semigroups [28], which is open source software.

The analogue of Cayley's Theorem for semigroups states that every finite semigroup is isomorphic to a transformation semigroup. Consequently, it could be argued that it is sufficient to have computational tools available for transformation semigroups only. An analogous argument could be made for arbitrary groups with respect to permutation groups, but developments in computational group theory suggest otherwise. For example, the Matrix Group Recognition Project has produced efficient algorithms for computing with groups of matrices over finite fields; [1] and [21]. Similarly, for some classes of semigroups, such as subsemigroups of the partition monoid or a Rees matrix semigroup, the only known faithful transformation representations are those that act on the elements of the semigroup itself. Hence, it is necessary in such examples to exhaustively compute the elements of the semigroup before a transformation representation is available. At this point any transformation representation, and the non-exhaustive methods that could be applied to it, are redundant. Therefore, to compute with such semigroups without exhaustively enumerating them, it is necessary to have non-exhaustive algorithms that apply directly to the given semigroups and, in particular, do not require a transformation representation. It is such algorithms that we present in this paper.

When considering matrix semigroups, it is straightforward to determine a transformation representation. Even in these cases, it is sometimes preferable to compute in the native matrix representation: in particular, where there are methods for matrix groups that are more efficient than computing a permutation representation.

The algorithms in this paper only apply to subsemigroups of a regular semigroup. However, it would be possible to modify several of these algorithms, including the main one (Algorithm 11), so that they apply to subsemigroups S of a non-regular semigroup U. In particular, if it were possible to determine whether a given element was regular or not in U, then we could use the data structure for $\mathscr{R}$-classes described in Section 5 for the regular elements, and perform an exhaustive enumeration of $\mathscr{R}$-classes of non-regular elements in U. Or alternatively, it might be possible to use a combination of the approaches described in this paper and those in [19, 20]. Such an approach would be possible with, say, semigroups of binary relations. It is possible to check that a binary relation is regular as an element of the semigroup of all binary relations in polynomial time; see [10, 37]. However, we did not yet follow this approach either in the paper or in the Semigroups package since it is relatively easy to find a transformation representation of a semigroup of binary relations.

This paper is organised as follows. In Section 2, we recall some well-known mathematical notions, and establish some notation that is required in subsequent sections. In Section 3, we provide the mathematical basis that proves the validity of the algorithms presented in Sections 5. In Section 4, we show that transformation semigroups, partial permutation semigroups and inverse semigroups, partition monoids, semigroups of matrices over a finite field, and subsemigroups of finite regular Rees matrix or 0-matrix semigroups satisfy the conditions from Section 3 and, hence, belong to the class of semigroups with which we can compute efficiently. Detailed algorithms are presented as pseudocode in Section 5, including some remarks about how the main algorithms presented can be simplified in the case of regular and inverse semigroups. In Section 6 we give several detailed examples.

2 Mathematical prerequisites

A semigroup S is *regular* if for every $x \in S$ there exists $x' \in S$ such that $xx'x = x$. A semigroup S is a *monoid* if it has an identity element, i.e. an element $e \in S$ such that $es = se = s$ for all $s \in S$. If S is a semigroup, we write S^1 for the monoid obtained by adjoining an identity 1_S to S if necessary.

For any set Ω, the set Ω^Ω of transformations of Ω is a semigroup under composition of functions, known as the *full transformation monoid on* Ω. The identity element of Ω^Ω is the identity function on Ω, which will be denoted id_Ω. We denote the full transformation monoid on the finite set $\{1, \ldots, n\}$ by T_n. Throughout this article, we will write functions to the right of their arguments and compose functions from left to right.

If X is a subset of a semigroup S, then the least subsemigroup of S containing X is denoted by $\langle X\rangle$; this is also referred to as the *subsemigroup generated by* X. We denote the cardinality of a set X by $|X|$.

Let S be a semigroup and let $x, y \in S$ be arbitrary. We say that x and y are $\mathscr{L}$-related if the principal left ideals generated by x and y in S are equal; in other words, $S^1x = S^1y$. We write $x\mathscr{L}y$ to denote that x and y are $\mathscr{L}$-related. In Section 3, we often want to distinguish between the cases when elements are $\mathscr{L}$-related in a semigroup U or in a subsemigroup S of U. We write $x\mathscr{L}^Sy$ or $x\mathscr{L}^Uy$ to differentiate these cases.

Green's $\mathscr{R}$-relation is defined dually to Green's $\mathscr{L}$-relation; Green's $\mathscr{H}$-relation is the meet, in the lattice of equivalence relations on S, of $\mathscr{L}$ and $\mathscr{R}$; and $\mathscr{D}$ is the join. We will refer to the equivalence classes as $\mathscr{K}$-classes where $\mathscr{K}$ is any of $\mathscr{R}$, $\mathscr{L}$, $\mathscr{H}$, or $\mathscr{D}$, and the $\mathscr{K}$-class of $x \in S$ will be denoted by K_x, or K_x^S if it is necessary to explicitly refer to the semigroup where the relation is defined. We denote the set of $\mathscr{K}$-classes of a semigroup S by $S/\mathscr{K}$.

In a finite semigroup, $x\mathscr{D}y$ if and only if the (2-sided) principal ideals generated by x and y are equal. Containment of principal ideals induces a partial order on the $\mathscr{D}$-classes of S, sometimes denoted $\leq_\mathscr{D}$; that is, $D_x \leq_\mathscr{D} D_y$ if and only if $S^1xS^1 \subseteq S^1yS^1$.

Proposition 2.1 (cf. Proposition A.1.16 in [36]). *Let U be a semigroup and let S be a subsemigroup of U. Suppose that x and y are regular elements of S. Then $x\mathscr{K}^Uy$ if and only if $x\mathscr{K}^Sy$, where $\mathscr{K}$ is any of $\mathscr{R}$, $\mathscr{L}$ or $\mathscr{H}$.*

Note that the previous result does not necessarily hold for $\mathscr{K} = \mathscr{D}$.

Let S be a semigroup and let Ω be a set. A function $\Psi : \Omega \times S^1 \longrightarrow \Omega$ is a *right action* of S on Ω if

- $((\alpha, s)\Psi, t)\Psi = (\alpha, st)\Psi$;

- $(\alpha, 1)\Psi = \alpha$.

For the sake of brevity, we will write $\alpha \cdot s$ instead of $(\alpha, s)\Psi$, and we will say that S acts on Ω on the right. *Left actions* are defined analogously, and we write $s \cdot \alpha$ in this case, and say S acts on Ω on the left. The *kernel* of a function $f : X \longrightarrow Y$, where X and Y are any sets, is the equivalence relation $\{(x, y) \in X \times X : (x)f = (y)f\}$. A right action of a semigroup S on a set Ω induces a homomorphism from S to the full transformation monoid on Ω defined by mapping $s \in S$ to the transformation defined by

$$\alpha \mapsto (\alpha, s)\Psi \quad \text{for all} \quad \alpha \in \Omega.$$

An action is called *faithful* if the induced homomorphism is injective. The *kernel of a right action* of a semigroup S on a set Ω is just the kernel of the induced homomorphism, i.e. the equivalence relation $\{(s, t) \in S \times S : \alpha \cdot s = \alpha \cdot t \ (\forall \alpha \in \Omega)\}$. The kernel of a left action is defined analogously.

If S acts on the sets Ω and Ω' on the right, then we say that $\lambda : \Omega \longrightarrow \Omega'$ is a *homomorphism of right actions* if $(\alpha \cdot s)\lambda = (\alpha)\lambda \cdot s$ for all $\alpha \in \Omega$ and $s \in S^1$. *Homomorphisms of left actions* are defined analogously. An *isomorphism* of (left or right) actions is a bijective homomorphism of (left or right) actions.

If Ω is a set, then we denote the set of subsets of Ω by $\mathcal{P}(\Omega)$. If S is a semigroup acting on the right on Ω, then the action of S on Ω induces a natural action of S on $\mathcal{P}(\Omega)$, which we write as:

$$\Sigma \cdot s = \{\alpha \cdot s : \alpha \in \Sigma\} \qquad \text{for each } \Sigma \subseteq \Omega. \tag{2.2}$$

We will denote the function from Σ to $\Sigma \cdot s$ defined by $\alpha \mapsto \alpha \cdot s$ by $s|_\Sigma$. We define the *stabiliser* of Σ under S to be

$$\mathrm{Stab}_S(\Sigma) = \{s \in S^1 : \Sigma \cdot s = \Sigma\}.$$

The quotient of the stabiliser by the kernel of its action on Σ, i.e. the congruence

$$\{(s, t) : s, t \in \mathrm{Stab}_S(\Sigma),\ s|_\Sigma = t|_\Sigma\},$$

is isomorphic to

$$S_\Sigma = \{s|_\Sigma : s \in \mathrm{Stab}_S(\Sigma)\}$$

which in the case that Σ is finite, is a subgroup of the symmetric group $\mathrm{Sym}(\Sigma)$ on Σ. The stabiliser S_Σ can also be seen as a subgroup of $\mathrm{Sym}(\Omega)$ by extending the action of its elements so that they fix $\Omega \setminus \Sigma$ pointwise. It is immediate that $s|_\Sigma \cdot t|_\Sigma = (st)|_\Sigma$ for all $s, t \in \mathrm{Stab}_S(\Sigma)$.

If S acts on Ω, the *strongly connected component* (usually abbreviated to s.c.c.) of an element $\alpha \in \Omega$ is the set of all $\beta \in \Omega$ such that $\beta = \alpha \cdot s$ and $\alpha = \beta \cdot t$ for some $s, t \in S^1$. We write $\alpha \sim \beta$ if α and β belong to the same s.c.c. and the action is clear from the context.

If S is not a group and $\alpha \in \Omega$, then

$$\alpha \cdot S^1 = \{\alpha \cdot s : s \in S^1\},$$

is a disjoint union of strongly connected components of the action of S. Note that $\alpha \cdot S^1$ might consist of more than one strongly connected component. If S is a group, then $\alpha \cdot S^1$ has only one strongly connected component, which is usually called the *orbit* of α under S.

Proposition 2.3. *Let $S = \langle X \rangle$ be a semigroup that acts on a finite set Ω on the right and let $\Sigma_1, \ldots, \Sigma_n \subseteq \Omega$ be the elements of a strongly connected component of the action of S on $\mathcal{P}(\Omega)$. Then the following hold:*

(a) *if $\Sigma_1 \cdot u_i = \Sigma_i$ for some $u_i \in S^1$, then there exists $\overline{u_i} \in S^1$ such that $\Sigma_i \cdot \overline{u_i} = \Sigma_1$, $(u_i\overline{u_i})|_{\Sigma_1} = \mathrm{id}_{\Sigma_1}$, and $(\overline{u_i}u_i)|_{\Sigma_i} = \mathrm{id}_{\Sigma_i}$;*

(b) *S_{Σ_i} and S_{Σ_j} are conjugate subgroups of $\mathrm{Sym}(\Omega)$ for all $i, j \in \{1, \ldots, n\}$;*

(c) *if $u_1 = \overline{u_1} = 1_S$ and $u_i, \overline{u_i} \in S$ are as in part (a) for $i > 1$, then S_{Σ_1} is generated by*

$$\{(u_i x \overline{u_j})|_{\Sigma_1} : 1 \leq i, j \leq n,\ x \in X,\ \Sigma_i \cdot x = \Sigma_j\}.$$

Proof. Let $\theta : S \longrightarrow \Omega^\Omega$ be the homomorphism induced by the action of S on Ω. Then $(S)\theta$ is a transformation semigroup and the actions of $(S)\theta$ and S on Ω are equal. Hence (a), (b), and (c) are just Lemma 2.2 and Theorems 2.1 and 2.3 in [23], respectively. $\square$

We will refer to the generators of S_{Σ_1} in Proposition 2.3(c) as *Schreier generators* of S_{Σ_1}, due to the similarity of this proposition and Schreier's Theorem [45, Theorem 2.57]. If S is a semigroup acting on a set Ω, if $\Sigma, \Gamma \subseteq \Omega$ are such that $\Sigma \sim \Gamma$ and if $u \in S$ is any element such that $\Sigma \cdot u = \Gamma$, then we write $\overline{u}$ to denote an element of S with the properties in Proposition 2.3(a).

The analogous definitions can be made, and an analogous version of Proposition 2.3 holds for left actions. In the next section there are several propositions involving left and right actions in the same statement, and so we define the following notation for left actions. If S is a semigroup acting on the left on a set Ω and $\Sigma \subseteq \Omega$, then the induced left action of S on $\mathcal{P}(\Omega)$ is defined analogously to (2.2); the function $\alpha \mapsto s \cdot \alpha$ is denoted by ${}_{\Sigma}|s$; and we define

$$ {}_{\Sigma}S = \{{}_{\Sigma}|s : s \cdot \Sigma = \Sigma\} $$

and in the case that Ω is finite, ${}_{\Sigma}S \leq \mathrm{Sym}(\Omega)$.

3 From transformation semigroups to arbitrary regular semigroups

In this section, we will generalise the results of [23] from transformation semigroups to subsemigroups of an arbitrary finite regular semigroup.

Generally speaking, the central notion is that for a fixed semigroup U with a determined structure, we can use the properties of U to produce algorithms to compute any subsemigroup S of U specified by a generating set. For example, U can be the full transformation monoid or the symmetric inverse monoid on a finite set. Roughly speaking, the subsemigroup S can be decomposed into its $\mathscr{R}$-classes, and an $\mathscr{R}$-class can be decomposed into the stabiliser S_L (under right multiplication on $\mathcal{P}(U)$) of an $\mathscr{L}$-class L in U and the s.c.c. of L under the action of S on the $\mathscr{L}$-classes of U. Decomposing S in this way will permit us to efficiently compute many aspects of the structure of S without enumerating its elements exhaustively. For instance, using this decomposition, we can test membership in S, compute the size, Green's structure, idempotents, elements, and maximal subgroups of S.

Throughout the remainder of this section we suppose that U is an arbitrary finite semigroup, and S is a subsemigroup of U.

3.1 Equivalent actions on Green's classes

We require the following actions of S: the action on $\mathcal{P}(U)$ induced by multiplying elements of U on the right by $s \in S^1$, i.e.:

$$ As = \{as : a \in A\} \tag{3.1} $$

where $s \in S^1$ and $A \subseteq U$; and the action of S on $U/\mathscr{L}$ defined by

$$ L_x \cdot s = L_{xs} \tag{3.2} $$

for all $x \in U$ and $s \in S^1$. The latter defines an action since $\mathscr{L}$ is a right congruence.

In general, the actions defined in (3.1) and (3.2) are not equal when restricted to $U/\mathscr{L}$. For example, it can be the case that $L_x s \subsetneq L_{xs}$ and, in particular, $L_x s \notin U/\mathscr{L}$. However, the actions do coincide in one case, as described in the next lemma, which is particularly important here.

Lemma 3.3. *Let $x, y \in U$ be arbitrary. Then $L_x, L_y \in U/\mathscr{L}$ belong to the same s.c.c. of the action of S defined by (3.2) if and only if L_x and L_y belong to the same s.c.c. of the action of S defined by (3.1).*

Proof. The converse implication is trivial. If $L_x, L_y \in U/\mathscr{L}$ belong to the same strongly connected component under the action (3.2), then, there exists $s \in S^1$ such that $L_{xs} = L_y$. Hence, by Green's Lemma [17, Lemma 2.2.1], the function from L_x to $L_{xs} = L_y$ defined by $z \mapsto zs$ for all $z \in L_x$ is a bijection and so $L_x \cdot s = L_{xs} = \{ys : y \in L_x\}$. □

Although S does not, in general, act on the $\mathscr{L}$-classes of U by right multiplication, by Lemma 3.3, it does act by right multiplication on the set of $\mathscr{L}$-classes within a strongly connected component of its action. We will largely be concerned with the strongly connected components of the restriction of the action on $\mathcal{P}(U)$ in (3.1) to $U/\mathscr{L}$. In this context, by Lemma 3.3, we may, without loss of generality, use the actions defined in (3.1) and (3.2) interchangeably.

In general, it is impractical to compute directly with the actions defined in (3.1) and (3.2). However, we can replace these actions by equivalent actions in the following sense.

Definition 3.4. We say that a right action of S on a set Ω is *equivalent (via λ)* to the action of S on U by right multiplication if there exists a homomorphism $\lambda : U \longrightarrow \Omega$ of these actions where the kernel of λ is $\mathscr{L}^U$, i.e. $\ker(\lambda) = \{(u, v) \in U \times U : (u)\lambda = (v)\lambda\} = \mathscr{L}^U$.

The following lemma justifies the use of the word "equivalent" in the previous definition.

Lemma 3.5. *Suppose that S has a right action on a set Ω that is equivalent via $\lambda : U \longrightarrow \Omega$ to the action of S on U by right multiplication. Then the following hold:*

(a) *if $x, y \in U$, then: $(x)\lambda \sim (y)\lambda$ if and only if $L_x \sim L_y$ under the action of S on $U/\mathscr{L}$ defined in (3.2);*

(b) *if Ω is the s.c.c. of an $\mathscr{L}^U$-class under the action of S on $\mathcal{P}(U)$ by right multiplication, then λ induces an isomorphism from the natural action of $\mathrm{Stab}_S(\Omega)$ on Ω to the action of $\mathrm{Stab}_S(\Omega)$ on $\{(x)\lambda : L^U_x \in \Omega\}$.*

Proof. **(a)** This follows immediately from the definition of a homomorphism of actions, and the assumption that the kernel of λ is $\mathscr{L}^U$.

(b) Let $X = \{(x)\lambda : L^U_x \in \Omega\}$ and let $\theta : \Omega \longrightarrow X$ be defined by $(L^U_x)\theta = (x)\lambda$. Since the kernel of λ is $\mathscr{L}^U$, it follows that θ is well-defined, and a bijection. If $L^U_x \in \Omega$ and $s \in \mathrm{Stab}_S(\Omega)$, then, by Lemma 3.3, it follows that

$$(L^U_x s)\theta = (L^U_x \cdot s)\theta = (L^U_{xs})\theta = (xs)\lambda,$$

and since λ is a homomorphism of actions:

$$(xs)\lambda = (x)\lambda \cdot s = (L^U_x)\theta \cdot s.$$

Thus θ is a isomorphism of the actions of $\mathrm{Stab}_S(\Omega)$ on Ω and X, as required. □

It follows from Lemma 3.5 that any statement about either of the actions of S defined in (3.1) or (3.2) within a strongly connected component of $\mathscr{L}^U$-classes can be replaced with an equivalent statement about the action of S within a strongly connected component of the action of S on Ω.

Throughout this section we suppose that S has a right action on some set Ω equivalent, via $\lambda : U \longrightarrow \Omega$, to the action of S on U by right multiplication. As an *aide-mémoire*, we will write $(U)\lambda$ or $(S)\lambda$ instead of Ω. We also write ρ to denote the analogue of λ for left actions. More precisely, we suppose that S has a left action on a set $(U)\rho$, the kernel of this action is $\mathscr{R}^U$, and there is a homomorphism $\rho : U \longrightarrow (U)\rho$ of the actions of S on U by left multiplication and of S on $(U)\rho$. In Section 4, we will show how to obtain λ from Definition 3.4, and its analogue ρ, for several well-known classes of semigroups, such that we can compute with the action of S on $(S)\lambda$ and $(S)\rho$ efficiently.

Recall that we write $\alpha \sim \beta$ to denote that α and β belong to the same strongly connected component of an action. We will make repeated use of the following lemma later in this section.

Lemma 3.6. *Let $x \in S$ and $s, t \in S^1$ be arbitrary. Then:*

(a) *$(x)\lambda \sim (xs)\lambda$ if and only if $x\mathscr{R}^S xs$;*

(b) *$(x)\rho \sim (tx)\rho$ if and only if $x\mathscr{L}^S tx$;*

(c) *$(x)\lambda \sim (xs)\lambda$ and $(x)\rho \sim (tx)\rho$ together imply that $x\mathscr{D}^S txs$.*

Proof. We only prove parts (a) and (c), since the proof of part (b) is dual to that of (a).

(a). ($\Rightarrow$) Suppose that $(x)\lambda \sim (xs)\lambda$. Then L^U_{xs} and L^U_x belong to the same s.c.c. of the action of S on $U/\mathscr{L}$ by right multiplication. Hence, by Proposition 2.3(a), there exist $\overline{s} \in S^1$ such that $L^U_{xs} \cdot \overline{s} = L^U_x$ and $(s\overline{s})|_{L^U_x} = \mathrm{id}_{L^U_x}$. Hence, in particular, $xs\overline{s} = x$ and so $xs\mathscr{R}^S x$.

($\Leftarrow$) Suppose $x\mathscr{R}^S xs$. Then there exists $t \in S^1$ such that $xst = x$. It follows that $(x)\lambda \cdot s = (xs)\lambda$ and $(xs)\lambda \cdot t = (x)\lambda$. Hence $(x)\lambda \sim (xs)\lambda$.

(c). Suppose that $(x)\lambda \sim (xs)\lambda$ and $(x)\rho \sim (tx)\rho$. Then, by parts (a) and (b), $x\mathscr{R}^S xs$ and $x\mathscr{L}^S tx$. The latter implies that $xs\mathscr{L}^S txs$, and so $x\mathscr{D}^S txs$. □

3.2 Faithful representations of stabilisers

Let U be an arbitrary finite semigroup, and let S be any subsemigroup of U.

In the same way that it is impractical to compute with the action of S on the $\mathscr{L}$-classes of U, it is equally impractical to compute directly with the natural action of the stabiliser of an $\mathscr{L}$-class in S on that $\mathscr{L}$-class. For example, the $\mathscr{L}$-class L of any transformation $x \in T_{10}$ with 5 points in its image has $5,103,000$ elements but, in this case, U_L has a faithful permutation representation on only 5 points.

With the preceding comments in mind, throughout this section we will make statements about arbitrary faithful representations of U_L, $L \in U/\mathscr{L}$, rather than about the natural action of U_L (or S_L) on L. More specifically, if $x \in U$ and ζ is any faithful representation of $U_{L_x^U}$, then we define

$$\mu_x : \mathrm{Stab}_U(L_x^U) \longrightarrow (U_{L_x^U})\zeta \qquad \text{by} \qquad (u)\mu_x = (u|_{L_x^U})\zeta \qquad \text{for all } u \in U. \tag{3.7}$$

It is clear that μ_x is a homomorphism. Since S is a subsemigroup of U, it follows that $\mathrm{Stab}_S(L_x^U)$ is a subsemigroup $\mathrm{Stab}_U(L_x^U)$ and $S_{L_x^U}$ is a subgroup of $U_{L_x^U}$. Hence $\mu_x : \mathrm{Stab}_U(L_x^U) \longrightarrow (U_{L_x^U})\zeta$ restricted to $\mathrm{Stab}_S(L_x^U)$ is a homomorphism from $\mathrm{Stab}_S(L_x^U)$ to $(S_{L_x^U})\zeta$.

It is possible that, since ζ depends of the $\mathscr{L}$-class of x in U, we should use $\zeta_{L_x^U}$ to denote the faithful representation given above. However, to simplify our notation we will not do this. Note that, with this definition, $(s,t) \in \ker(\mu_x)$ if and only if $s|_{L_x^U} = t|_{L_x^U}$. To simplify our notation, we will write S_x and U_x to denote $(\mathrm{Stab}_S(L_x^U))\mu_x = (S_{L_x^U})\zeta$ and $(\mathrm{Stab}_U(L_x^U))\mu_x = (U_{L_x^U})\zeta$, respectively. Analogously, for every $x \in U$ we suppose that we have a homomorphism ν_x from $\mathrm{Stab}_U(R_x^U)$ into a group where the image of ν_x is isomorphic to ${}_{R_x^U}U$. We write ${}_xS$ and ${}_xU$ for $(\mathrm{Stab}_S(R_x^U))\nu_x$ and $(\mathrm{Stab}_U(R_x^U))\nu_x$, respectively.

In the case that S is a transformation, partial permutation, matrix, or partition semigroup, or a subsemigroup of a Rees 0-matrix semigroup, we will show in Section 4 how to obtain faithful representations of the stabilisers of $\mathscr{L}$- and $\mathscr{R}$-classes as permutation or matrix groups. It is then possible to use algorithms from Computational Group Theory to compute with these groups.

We will make repeated use of the following straightforward lemma.

Lemma 3.8. *Let $x \in U$ and $s, t \in \mathrm{Stab}_U(L_x^U)$ be arbitrary. Then the following are equivalent:*

(a) *$(s)\mu_x = (t)\mu_x$;*

(b) *$ys = yt$ for all $y \in L_x^U$;*

(c) *there exists $y \in L_x^U$ such that $ys = yt$.*

Proof. (a) ⇒ (b) If $(s)\mu_x = (t)\mu_x$, then $(s|_{L_x^U})\zeta = (t|_{L_x^U})\zeta$ and so $s|_{L_x^U} = t|_{L_x^U}$. It follows that $ys = yt$ for all $y \in L_x^U$.

(b) ⇒ (c) is trivial.

(c) ⇒ (a) Suppose that $y \in L_x^U$ is such that $ys = yt$. If $z \in L_x^U$ is arbitrary, then there exists $u \in U^1$ such that $z = uy$. Hence $zs = uys = uyt = zt$ and so $s|_{L_x^U} = t|_{L_x^U}$. It follows, by the definition of μ_x, that $(s,t) \in \ker(\mu_x)$ and so $(s)\mu_x = (t)\mu_x$. □

The analogue of Lemma 3.8 also holds for $\mathrm{Stab}_U(R_x^U)$ and ν_x; the details are omitted.

3.3 A decomposition for Green's classes

In this section, we show how to decompose an $\mathscr{R}$- or $\mathscr{L}$-class of our subsemigroup S as briefly discussed above. Recall that we supposed that S has a right action on some set equivalent via $\lambda : U \longrightarrow (U)\lambda$ to the action of S on U by right multiplication (Definition 3.4).

Proposition 3.9 (cf. Theorems 3.3 and 4.3 in [23]). *If $x \in S$ is arbitrary, then:*

(a) *$\{(y)\lambda : y\mathscr{R}^S x\}$ is a strongly connected component of the right action of S on $(S)\lambda$;*

(b) *$\{(y)\rho : y\mathscr{L}^S x\}$ is a strongly connected component of the left action of S on $(S)\rho$.*

Proof. We only prove the first statement, as the proof of the second is dual. Suppose $y \in S$ and $x \neq y$. Then $y\mathscr{R}^S x$ if and only if there exists $s \in S$ such that $xs = y$ and $xs\mathscr{R}^S x$. By Lemma 3.6(a), $xs\mathscr{R}^S x$ if and only if $(x)\lambda \sim (xs)\lambda$. □

Corollary 3.10. *Let $x, y, s, t \in S$. Then the following hold:*

(a) *if $x \mathscr{R}^S y$ and $xs \mathscr{L}^U y$, then $xs \mathscr{R}^S y$;*

(b) *if $x \mathscr{L}^S y$ and $tx \mathscr{R}^U y$, then $tx \mathscr{L}^S y$.*

Proof. Again, we only prove part (a). Since $xs \mathscr{L}^U y$, it follows that $(xs)\lambda = (y)\lambda$ and, since $x \mathscr{R}^S y$, by Proposition 3.9, $(x)\lambda \sim (y)\lambda = \lambda(xs)$. Hence $xs \mathscr{R}^S x$ by Lemma 3.6, and, since $x \mathscr{R}^S y$ by assumption, the proof is complete. □

Proposition 3.11 (cf. Theorems 3.7 and 4.6 in [23])**.** *Suppose that $x \in S$ and there exists $x' \in U$ where $xx'x = x$ (i.e. x is regular in U). Then the following hold:*

(a) *$L_x^U \cap R_x^S = \{y \in R_x^S : (y)\lambda = (x)\lambda\}$ is a group under the multiplication $*$ defined by $s * t = sx't$ for all $s, t \in L_x^U \cap R_x^S$ and its identity is x;*

(b) *$\phi : S_x \longrightarrow L_x^U \cap R_x^S$ defined by $((s)\mu_x)\phi = xs$, for all $s \in \operatorname{Stab}_S(L_x^U)$, is an isomorphism;*

(c) *$\phi^{-1} : L_x^U \cap R_x^S \longrightarrow S_x$ is defined by $(s)\phi^{-1} = (x's)\mu_x$ for all $s \in L_x^U \cap R_x^S$.*

Proof. We begin by showing that x is an identity under the multiplication $*$ of $L_x^U \cap R_x^S$. Since $x'x \in L_x^U$ and $xx' \in R_x^U$ are idempotents, it follows that $x'x$ is a right identity for L_x^U and xx' is a left identity for $R_x^S \subseteq R_x^U$. So, if $s \in L_x^U \cap R_x^S$ is arbitrary, then

$$x * s = xx's = s = sx'x = s * x,$$

as required.

We will prove that part (b) holds, which implies part (a).

ϕ is well-defined. If $s \in \operatorname{Stab}_S(L_x^U)$, then $xs \mathscr{L}^U x$. Hence, by Corollary 3.10(a), $xs \mathscr{R}^S x$ and so $((s)\mu_x)\phi = xs \in L_x^U \cap R_x^S$. If $t \in \operatorname{Stab}_S(L_x^U)$ is such that $(t)\mu_x = (s)\mu_x$, then, by Lemma 3.8, $xt = xs$.

ϕ is surjective. Let $s \in L_x^U \cap R_x^S$ be arbitrary. Then $xx's = x * s = s$ since x is the identity of $L_x^U \cap R_x^S$. It follows that

$$L_x^U \cdot x's = L_s^U = L_x^U$$

and so $x's \in \operatorname{Stab}_U(L_x^U)$. Since $x \mathscr{R}^S s$, there exists $u \in S^1$ such that $xu = s = xx's$. It follows that $u \in \operatorname{Stab}_S(L_x^U)$ and, by Lemma 3.8, $(u)\mu_x = (x's)\mu_x$. Thus $((u)\mu_x)\phi = xu = s$ and ϕ is surjective.

ϕ is a homomorphism. Let $s, t \in \operatorname{Stab}_S(L_x^U)$. Then, since $xs \in L_x^U$ and $x'x$ is a right identity for L_x^U,

$$((s)\mu_x)\phi * ((t)\mu_x)\phi = xs * xt = xsx'xt = xst = ((st)\mu_x)\phi = ((s)\mu_x \cdot (t)\mu_x)\phi,$$

as required.

ϕ is injective. Let $\theta : L_x^U \cap R_x^S \longrightarrow S_x$ be defined by $(y)\theta = (x'y)\mu_x$ for all $y \in L_x^U \cap R_x^S$. We will show that $\phi\theta$ is the identity mapping on S_x, which implies that ϕ is injective, that $(y)\theta \in S_x$ for all $y \in L_x^U \cap R_x^S$ (since ϕ is surjective), and also proves part (c) of the proposition. If $s \in \operatorname{Stab}_S(L_x^U)$, then $((s)\mu_x)\phi\theta = (xs)\theta = (x'xs)\mu_x$. But $xx'xs = xs$ and so $(x'xs)\mu_x = (s)\mu_x$ by Lemma 3.8. Therefore, $((s)\mu_x)\phi\theta = (s)\mu_x$, as required. □

We state the analogue of Proposition 3.11 for the action of S on $U/\mathscr{R}$ by left multiplication.

Proposition 3.12. *Suppose that $x \in S$ and there exists $x' \in U$ where $xx'x = x$. Then the following hold:*

(a) *$L_x^S \cap R_x^U = \{y \in L_x^S : (y)\rho = (x)\rho\}$ is a group under the multiplication $*$ defined by $s * t = sx't$ for all $s, t \in L_x^S \cap R_x^U$ and its identity is x;*

(b) *$\phi : {}_xS \longrightarrow L_x^S \cap R_x^U$ defined by $((s)\nu_x)\phi = sx$, for all $s \in \operatorname{Stab}_S(R_x^U)$, is an isomorphism;*

(c) *$\phi^{-1} : L_x^S \cap R_x^U \longrightarrow {}_xS$ is defined by $(s)\phi^{-1} = (sx')\nu_x$ for all $s \in L_x^S \cap R_x^U$.*

We can also characterise an $\mathscr{H}$-class in a subsemigroup of a semigroup in terms of the stabilisers of its $\mathscr{L}$- and $\mathscr{R}$-class. Note that in the special case that $S = U$, it follows immediately from Proposition 3.11 that U_x is isomorphic to $H_x^U = L_x^U \cap R_x^U$ under the operation $*$ defined in the proposition.

Proposition 3.13 (cf. Theorem 5.1 in [23]). *Suppose that $x \in S$ and there exists $x' \in U$ where $xx'x = x$. Then the following hold:*

(a) *$\Psi : {}_xS \longrightarrow U_x$ defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$, $s \in \mathrm{Stab}_S(R^U_x)$, is an embedding;*

(b) *H^S_x is a group under the multiplication $s * t = sx't$, with identity x, and it is isomorphic to $S_x \cap ({}_xS)\Psi$;*

(c) *H^S_x under $*$ is isomorphic to the Schützenberger group of H^S_x.*

Proof. **(a).** Let $\phi : {}_xS \longrightarrow L^S_x \cap R^U_x$ be the isomorphism defined in Proposition 3.12(b), and let $\theta : L^U_x \cap R^U_x \longrightarrow U_x$ be the isomorphism defined in Proposition 3.11(c) (applied to U as a subsemigroup of itself). Then, since $L^S_x \cap R^U_x \subseteq L^U_x \cap R^U_x$, $\phi\theta : {}_xS \longrightarrow U_x$ is a embedding (being the composition of injective homomorphisms). By definition, $((s)\nu_x)\phi\theta = (x'sx)\mu_x = ((s)\nu_x)\Psi$, for all $s \in \mathrm{Stab}_S(R^U_x)$.

(b). Note that $H^S_x = L^S_x \cap R^S_x = (L^U_x \cap R^S_x) \cap (R^U_x \cap L^S_x)$. From Proposition 3.11(c), $\phi_1^{-1} : R^S_x \cap L^U_x \longrightarrow S_x \leq U_x$ defined by

$$(s)\phi_1^{-1} = (x's)\mu_x$$

is an isomorphism (where $R^S_x \cap L^U_x$ has multiplication $*$ defined in Proposition 3.11(a) as $s * t = sx't$ for all $s, t \in R^S_x \cap L^U_x$). Similarly, by Proposition 3.12(c), $\phi_2^{-1} : R^U_x \cap L^S_x \longrightarrow {}_xS$ defined by

$$(s)\phi_2^{-1} = (sx')\nu_x$$

is an isomorphism. Hence if $s \in H^S_x$, then, by Proposition 3.11(a),

$$(s)\phi_2^{-1}\Psi = (x'sx'x)\mu_x = (x's)\mu_x = (s)\phi_1^{-1}$$

and so ϕ_1^{-1}, restricted to H^S_x, is an injective homomorphism from H^S_x under $*$ into $S_x \cap ({}_xS)\Psi$.

If $g \in S_x \cap ({}_xS)\Psi$, then there exists $a \in \mathrm{Stab}_S(R^U_x)$ such that $(x'ax)\mu_x = g$. Since $ax \mathscr{R}^U x$ and xx' is a left identity in R^U_x, it follows that $xx'ax = ax$ and so $ax = xx'ax = ((x'ax)\mu_x)\phi_1 \in R^S_x \cap L^U_x \subseteq R^S_x$ where ϕ_1 is given in Proposition 3.11(b). Similarly, $(a)\nu_x \in {}_xS$ implies that $ax = ((a)\nu_x)\phi_2 \in R^U_x \cap L^S_x \subseteq L^S_x$. Therefore $ax \in H^S_x$ and $(ax)\phi_1^{-1} = (x'ax)\mu_x = g$, and so ϕ_1^{-1} is surjective and thus an isomorphism from H^S_x to $S_x \cap ({}_xS)\Psi$, as required.

(c). The *Schützenberger group* of $H = H^S_x$ is defined to be the quotient of $\mathrm{Stab}_S(H)$ by the kernel of its action on H (by right multiplication on the elements of H). In our notation the Schützenberger group of H is denoted S_H, however in the literature it is usually denoted by $\Gamma_R(H)$. It is well-known that $\Gamma_R(H)$ acts transitively and freely on H, see for example [36, Section A.3.1]. It follows that $\phi : \Gamma_R(H) \longrightarrow H$ defined by

$$(s|_H)\phi = xs$$

is a bijection. If $s|_H, t|_H \in \Gamma_R(H)$, then $xs \in H$ and so $xsx'x = xs * x = xs$, since x is the identity of H by Proposition 3.11(a). Thus

$$(s|_H \cdot t|_H)\phi = xst = xsx'xt = (s|_H)\phi * (t|_H)\phi$$

and ϕ is an isomorphism, as required. □

The statement of Proposition 3.13 can be simplified somewhat in the case that the element $x \in S$ is regular in S and not only in U.

Corollary 3.14. *Suppose that $x \in S$ and there exists $x' \in S$ where $xx'x = x$. Then the following three groups are isomorphic: H^S_x under the multiplication $s * t = sx't$, S_x, and ${}_xS$. Furthermore, $\Psi : {}_xS \longrightarrow S_x$ defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$ is an isomorphism.*

Proof. Since x is regular in S, it follows that $H^S_x = L^S_x \cap R^S_x = L^U_x \cap R^S_x$ and, similarly, $H^S_x = R^U_x \cap L^S_x$. So, the first part of the statement follows by Propositions 3.11(b) and 3.12(b).

By Propositions 3.12(b) and 3.11(c), respectively, there exist isomorphisms $\phi : {}_xS \longrightarrow L^S_x \cap R^U_x$ and $\theta : L^U_x \cap R^S_x \longrightarrow S_x$. Therefore since $H^S_x = L^U_x \cap R^S_x = R^U_x \cap L^S_x$, it follows that $\Psi = \phi\theta : {}_xS \longrightarrow S_x$ is an isomorphism. □

We collect some corollaries of what we have proved so far.

Corollary 3.15. *If $x, y \in S$ are regular elements of U, then the following hold:*

(a) *If $x\mathscr{R}^S y$, then $S_{L_x^U}$ and $S_{L_y^U}$ are conjugate subgroups of* $\mathrm{Sym}(U)$*. In particular, S_x and S_y are isomorphic;*

(b) *$|R_x^S|$ equals the size of the group S_x multiplied by the size of the s.c.c. of $(x)\lambda$ under the action of S on $(S)\lambda$;*

(c) *If $(x)\lambda \sim (y)\lambda$, then $|R_x^S| = |R_y^S|$;*

(d) *If $x\mathscr{R}^S y$ and $u \in S^1$ is such that $(x)\lambda \cdot u = (y)\lambda$, then the function from $L_x^U \cap R_x^S$ to $L_y^U \cap R_x^S$ defined by $s \mapsto su$ is a bijection.*

Proof. **(a).** Since $x\mathscr{R}^S y$, it follows by Proposition 3.9(a) that L_x^U and L_y^U are in the same s.c.c. of the action of S on the $\mathscr{L}$-classes of U. Thus, by Proposition 2.3(b), it follows that $S_{L_x^U}$ and $S_{L_y^U}$ are conjugate subgroups of the symmetric group on U, and so, in particular, are isomorphic.

(b). The set R_x^S is partitioned by the sets $R_x^S \cap L_y^U = R_y^S \cap L_y^U$ for all $y \in R_x^S$. By Proposition 3.11(b), $|R_y^S \cap L_y^U| = |S_y|$ and by part (a), $|S_y| = |S_x|$ for all $y \in R_x^S$. Thus $|R_x^S|$ equals the number of distinct values of λ when applied to elements of R_x^S multiplied by $|S_x|$. Proposition 3.9(a) says that $\{(y)\lambda : y \in R_x^S\}$ is a s.c.c. of the right action of S on $(S)\lambda$.

(c). This follows immediately from parts (a) and (b).

(d). Let $s \in L_x^U \cap R_x^S$ be arbitrary. Then $s\mathscr{R}^S x\mathscr{R}^S y$ and $su\mathscr{L}^U xu\mathscr{L}^U y$, and so, by Corollary 3.10(a), $su\mathscr{R}^S y$, i.e. $su \in L_y^U \cap R_x^S$. By Proposition 2.3(a), there exists $\overline{u} \in S^1$ such that $su\overline{u} = s$ for all $s \in L_x^U \cap R_x^S$. Therefore $s \mapsto su$ and $t \mapsto t\overline{u}$ are mutually inverse bijections from $L_x^U \cap R_x^S$ to $L_y^U \cap R_x^S$ and back. $\square$

For the sake of completeness, we state the analgoue of Corollary 3.15 for $\mathscr{L}$-classes.

Corollary 3.16. *If $x, y \in S$ are regular elements of U, then the following hold:*

(a) *If $x\mathscr{L}^S y$, then ${}_{R_x^U}S$ and ${}_{R_y^U}S$ are conjugate subgroups of* $\mathrm{Sym}(U)$*. In particular, ${}_xS$ and ${}_yS$ are isomorphic;*

(b) *$|L_x^S|$ equals the size of the group ${}_xS$ multiplied by the size of the s.c.c. of $(x)\rho$ under the action of S on $(S)\rho$;*

(c) *If $(x)\rho \sim (y)\rho$, then $|L_x^S| = |L_y^S|$;*

(d) *If $x\mathscr{L}^S y$ and $u \in S^1$ is such that $u \cdot (x)\rho = (y)\rho$, then the function from $R_x^U \cap L_x^S$ to $R_y^U \cap L_x^S$ defined by $s \mapsto us$ is a bijection.*

3.4 Membership testing

Let U be a semigroup and let S be a subsemigroup of U. The next proposition shows that testing membership in an $\mathscr{R}$-class of S is equivalent to testing membership in a stabiliser of an $\mathscr{L}$-class of U. Since the latter is a group, this reduces the problem of membership testing in $\mathscr{R}$-classes to that of membership testing in a group, so we can then take advantage of efficient algorithms from computational group theory; such as the Schreier-Sims algorithm [45, Section 4.4].

Proposition 3.17. *Suppose that $x \in S$ and there is $x' \in U$ with $xx'x = x$. If $y \in U$ is arbitrary, then $y\mathscr{R}^S x$ if and only if $y\mathscr{R}^U x$, $(y)\lambda \sim (x)\lambda$, and $(x'yv)\mu_x \in S_x$ where $v \in S^1$ is any element such that $(y)\lambda \cdot v = (x)\lambda$.*

Proof. ($\Rightarrow$) Since $R_x^S \subseteq R_x^U$, $y\mathscr{R}^U x$ and from Proposition 3.9(a), $(y)\lambda \sim (x)\lambda$. Suppose that $v \in S^1$ is such that $(y)\lambda \cdot v = (x)\lambda$. Then, by Corollary 3.15(d), $yv \in L_x^U \cap R_x^S$ and so, by Proposition 3.11(c), $(x'yv)\mu_x \in S_x$.

($\Leftarrow$) Since $y \in R_x^U$ and xx' is a left identity in its $\mathscr{R}^U$-class, it follows that $xx'y = y$. Suppose that $v \in S^1$ is any element such that $(y)\lambda \cdot v = (x)\lambda$ (such an element exists by assumption). Then, by assumption, $(x'yv)\mu_x \in S_x$ and so by Proposition 3.11(b), $yv = x \cdot x'yv \in L_x^U \cap R_x^S$. But $(y)\lambda \sim (yv)\lambda$, and so, by Lemma 3.6(a), $y\mathscr{R}^S yv$, and so $x\mathscr{R}^S yv\mathscr{R}^S y$, as required. $\square$

The following corollary of Proposition 3.17 will be important in Section 5.

Corollary 3.18. *Let $x \in S$ be such that there is $x' \in U$ with $xx'x = x$, and let $y \in U$ be such that there exist $u \in S^1$ and $v \in U^1$ with $(x)\lambda \cdot u = (y)\lambda$ and $xuv = x$. Then $y \mathscr{R}^S x$ if and only if $y \mathscr{R}^U x$, $(y)\lambda \sim (x)\lambda$, and $(x'yv)\mu_x \in S_x$.*

Proof. ($\Rightarrow$) That $y\mathscr{R}^U x$ and $(y)\lambda \sim (x)\lambda$ follows from Proposition 3.17. Suppose $u \in S$ and $v \in U$ are such that $(x)\lambda \cdot u = (y)\lambda$ and $xuv = x$. By Proposition 2.3(a) there exists $\overline{u} \in S^1$ such that $xu\overline{u} = x = xuv$. It follows that $z\overline{u} = zv$ for all $z \in L^S_{xu}$, and, in particular, $y\overline{u} = yv$. Hence, by Proposition 3.17, $(x'yv)\mu_x = (x'y\overline{u})\mu_x \in S_x$.

($\Leftarrow$) It suffices by Proposition 3.17 to show that there exists $w \in S^1$ such that $(y)\lambda \cdot w = (x)\lambda$ and $(x'yw)\mu_x \in S_x$. Since $(x)\lambda \sim (y)\lambda$ and $(x)\lambda \cdot u = (y)\lambda$, by Proposition 2.3(a) and Lemma 3.8, there exists $\overline{u} \in S^1$ such that $xu\overline{u} = x = xuv$. Hence, as above, $y\overline{u} = yv$ and so $(x'y\overline{u})\mu_x = (x'yv)\mu_x \in S_x$, as required. □

We state an analogue of Proposition 3.17 for $\mathscr{L}$-classes, with a slight difference.

Proposition 3.19. *Suppose that $x \in S$ and there is $x' \in U$ with $xx'x = x$. If $y \in U$ is arbitrary, then $y\mathscr{L}^S x$ if and only if $y \mathscr{L}^U x$, $(y)\rho \sim (x)\rho$, and $(x'vy)\mu_x \in ({}_xS)\Psi$ where $v \in S^1$ is any element such that $v \cdot (y)\rho = (x)\rho$ and $\Psi : {}_xS \longrightarrow U_x$ defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$ is the embedding from Proposition 3.13(a).*

Proof. The direct analogue of Proposition 3.17 states that $y\mathscr{L}^S x$ if and only if $y\mathscr{L}^U x$, $(y)\rho \sim (x)\rho$, and $(vyx')\nu_x \in {}_xS$. The last part is equivalent to $((vyx')\nu_x)\Psi = (x'vyx'x)\mu_x = (x'vy)\mu_x \in ({}_xS)\Psi$, as required. □

Propositions 3.17 and 3.19 allow us to express the elements of an $\mathscr{R}^S$- and $\mathscr{L}^S$-class in a particular form, which will be of use in the algorithms later in the paper.

Corollary 3.20. *Suppose that $x \in S$ and there is $x' \in U$ with $xx'x = x$. Then*

(a) *if $\mathcal{U}$ is any subset of S^1 such that $\{(xu)\lambda : u \in \mathcal{U}\} = \{(y)\lambda : (y)\lambda \sim (x)\lambda\}$, then*

$$R^S_x = \{xsu : s \in \mathrm{Stab}_S(L^U_x),\ u \in \mathcal{U}\};$$

(b) *if $\mathcal{V}$ is any subset of S^1 such that $\{(vx)\rho : v \in \mathcal{V}\} = \{(y)\rho : (y)\rho \sim (x)\rho\}$, then*

$$L^S_x = \{vtx : t \in \mathrm{Stab}_S(R^U_x),\ v \in \mathcal{V}\}.$$

Proof. We only prove part (a), since the proof of part (b) is dual.

Let $s \in \mathrm{Stab}_S(L^U_x)$ and let $u \in \mathcal{U}$ be arbitrary. Since $(xsu)\lambda = (xu)\lambda \sim (x)\lambda$, it follows, by Lemma 3.6(a), that $xsu\mathscr{R}^S x$.

If $y\mathscr{R}^S x$, then, by Proposition 3.9, $(y)\lambda \sim (x)\lambda$, and so there exists $u \in \mathcal{U}$ such that $(x)\lambda \cdot u = (y)\lambda$. Since xx' is a left identity for R^U_x, $xx'y = y$. By Proposition 2.3(a), there is $\overline{u} \in S^1$ such that $(y)\lambda \cdot \overline{u} = (x)\lambda$ and $y\overline{u}u = y$. Hence, by Proposition 3.17, $x'y\overline{u} \in \mathrm{Stab}_S(L^U_x)$ and $y = x \cdot x'y\overline{u} \cdot u$. □

We will prove the analogue of Proposition 3.17 for $\mathscr{D}$-classes, for which we require following proposition.

Proposition 3.21 (cf. Theorem 6.2 in [22])**.** *If $x \in S$ is such that there is $x' \in U$ with $xx'x = x$, then $D^S_x \cap H^U_x = \{sxt : s \in \mathrm{Stab}_S(R^U_x),\ t \in \mathrm{Stab}_S(L^U_x)\}$.*

Proof. Let $s \in \mathrm{Stab}_S(R^U_x)$ and $t \in \mathrm{Stab}_S(L^U_x)$ be arbitrary. It follows that $(xt)\lambda = (x)\lambda$ and $(sx)\rho = (x)\rho$, and so, by Lemma 3.6, $xt\mathscr{R}^S x$, $sx\mathscr{L}^S x$, and $x\mathscr{D}^S sxt$. Also $xt\mathscr{R}^S x$ implies that $sxt\mathscr{R}^S sx\mathscr{R}^U x$ and $sx\mathscr{L}^S x$ implies $sxt\mathscr{L}^S xt\mathscr{L}^U x$, and so $sxt \in H^U_x$, as required.

For the other inclusion, let $y \in D^S_x \cap H^U_x$ be arbitrary. Then $x\mathscr{D}^S y$ and so there is $s \in S^1$ such that $sx\mathscr{L}^S x$ and $sx\mathscr{R}^S y$. Hence $s \cdot R^U_x = R^U_{sx} = R^U_y = R^U_x$ and so $s \in \mathrm{Stab}_S(R^U_x)$. Since $sx\mathscr{R}^S y$, there exists $t \in S^1$ such that $sxt = y$ and so $L^U_x \cdot t = L^U_{sx} \cdot t = L^U_y = L^U_x$, which implies that $t \in \mathrm{Stab}_S(L^U_x)$, as required. □

Proposition 3.22. *Suppose that $x \in S$ and there is $x' \in U$ with $xx'x = x$. If $y \in U$ is arbitrary, then $y\mathscr{D}^S x$ if and only if $(y)\lambda \sim (x)\lambda$, $(y)\rho \sim (x)\rho$, and for any $u, v \in S^1$ such that $(y)\lambda \cdot u = (x)\lambda$ and $v \cdot (y)\rho = (x)\rho$ there exists $t \in \mathrm{Stab}_S(L^U_x)$ such that*

$$(x'vyu)\mu_x \cdot ((t)\mu_x)^{-1} \in ({}_xS)\Psi,$$

where $\Psi : {}_xS \longrightarrow U_x$, defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$, is the embedding from Proposition 3.13(a).

Proof. ($\Rightarrow$) Let $y \in D_x^S$. Then there exists $w \in S$ such that $y\mathscr{R}^S w\mathscr{L}^S x$. By Proposition 3.9(a) and (b), respectively, it follows that $(y)\lambda \sim (w)\lambda = (x)\lambda$, and $(x)\rho \sim (w)\rho = (y)\rho$.

Suppose that $u, v \in S^1$ are any elements such that $(y)\lambda \cdot u = (x)\lambda$ and $v \cdot (y)\rho = (x)\rho$. Then, by Lemma 3.6, $vy\mathscr{L}^S y$ and $yu\mathscr{R}^S y$, and so $vyu\mathscr{L}^S yu\mathscr{L}^U x$ and $vyu\mathscr{R}^S vy\mathscr{R}^U x$. Thus $vyu\mathscr{H}^U x$ and, since $y\mathscr{R}^S yu\mathscr{L}^S vyu$, it follows that $vyu\mathscr{D}^S x$.

By Proposition 3.21, there exist $s \in \mathrm{Stab}_S(R_x^U)$ and $t \in \mathrm{Stab}_S(L_x^U)$ such that $vyu = sxt$. Since $s \in \mathrm{Stab}_S(R_x^U)$, it follows that $((s)\nu_x)\Psi = (x'sx)\mu_x \in ({}_xS)\Psi$ (by Proposition 3.13(a)). In particular, $x'sx \in \mathrm{Stab}_U(L_x^U)$ and so $(x'vyu)\mu_x = (x'sxt)\mu_x = (x'sx)\mu_x \cdot (t)\mu_x$ and so

$$(x'vyu)\mu_x \cdot ((t)\mu_x)^{-1} = (x'sx)\mu_x \in ({}_xS)\Psi,$$

as required.

($\Leftarrow$) Suppose that $u, v \in S^1$ are any elements such that $(y)\lambda \cdot u = (x)\lambda$ and $v \cdot (y)\rho = (x)\rho$. Since $(y)\lambda \sim (x)\lambda = (yu)\lambda$ and $(y)\rho \sim (x)\rho = (vy)\rho$, it follows from Lemma 3.6(c), that $y\mathscr{D}^S vyu$. By assumption, there exist $s \in \mathrm{Stab}_S(R_x^U)$, $t \in \mathrm{Stab}_S(L_x^U)$ such that

$$(x'vyu)\mu_x = ((s)\nu_x)\Psi \cdot (t)\mu_x = (x'sxt)\mu_x.$$

In particular, by Lemma 3.8, $xx'vyu = xx'sxt$. Since xx' is a left identity for $R_x^U = R_{vy}^U$, we deduce that $xx'vy = vy$. Also, by Proposition 3.21, $sxt \in D_x^S \cap H_x^U$ implies that $R_{sxt}^U = R_x^U$ and so $xx'sxt = sxt$. Thus $y\mathscr{D}^S vyu = xx'vyu = xx'sxt = sxt\mathscr{D}^S x$. $\square$

3.5 Classes within classes

The next two propositions allow us to determine the $\mathscr{R}^S$-, $\mathscr{L}^S$- and $\mathscr{H}^S$-classes within a given $\mathscr{D}^S$-class in terms of the groups S_x, ${}_xS$, and the action of S on $(S)\lambda$ and $(S)\rho$.

If G is a group and H is a subgroup of G, then a *left transversal* of H in G is a set of left coset representatives of H in G. *Right transversals* are defined analogously.

Proposition 3.23. *Suppose that $x \in S$ and there is $x' \in U$ with $xx'x = x$ and that:*

(a) *$\mathcal{C}$ is a minimal subset of $\mathrm{Stab}_U(L_x^U)$ such that $\{(c)\mu_x : c \in \mathcal{C}\}$ is a left transversal of $S_x \cap ({}_xS)\Psi$ in $({}_xS)\Psi$ where $\Psi : {}_xS \longrightarrow U_x$, defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$, is the embedding from Proposition 3.13(a);*

(b) *$\{u_1, \ldots, u_m\}$ is a minimal subset of S^1 such that $\{u_i \cdot (x)\rho : 1 \le i \le m\}$ equals the s.c.c. of $(x)\rho$ under the left action of S on $(S)\rho$.*

Then $\{u_ixc : c \in \mathcal{C},\ 1 \le i \le m\}$ is a minimal set of $\mathscr{H}^S$-class representatives of L_x^S, and hence a minimal set of $\mathscr{R}^S$-class representatives for D_x^S.

Proof. We start by proving that for all $c \in \mathcal{C}$, there exists $c^* \in \mathrm{Stab}_S(R_x^U)$ such that $c^*x = xc$ and that $xc\mathscr{L}^S x$. Suppose that $c \in \mathcal{C}$ is arbitrary. Then $(c)\mu_x \in ({}_xS)\Psi$ and so there exists $c^* \in \mathrm{Stab}_S(R_x^U)$ such that $((c^*)\nu_x)\Psi = (x'c^*x)\mu_x = (c)\mu_x$. Hence, by Lemma 3.8, $xx'c^*x = xc$. Since $c^* \in \mathrm{Stab}_S(R_x^U)$, Proposition 3.12(b) implies that $c^*x\mathscr{R}^U x$ and $c^*x\mathscr{L}^S x$, and so $c^*x = xx'c^*x = xc$. It follows that $xc = c^*x\mathscr{L}^S x$.

By the assumption in part (b) and by Lemma 3.6(b), $u_ix\mathscr{L}^S x$ for all i, and so $u_ixc\mathscr{L}^S xc\mathscr{L}^S x$ for all i. Hence it suffices to show that $\{u_ixc : c \in \mathcal{C},\ 1 \le i \le m\}$ is a minimal set of $\mathscr{R}^S$-class representatives for D_x^S. In other words, if $y\mathscr{D}^S x$, then we must show that $y\mathscr{R}^S u_ixc$ for some $i \in \{1, \ldots, m\}$ and $c \in \mathcal{C}$, and that $(u_ixc, u_jxd) \notin \mathscr{R}^S$ if $i \neq j$ or $c \neq d$.

We start by showing that for every $y \in D_x^S \cap H_x^U$ there is $c \in \mathcal{C}$ such that $y\mathscr{R}^S xc$. By Proposition 3.21, there exist $s \in \mathrm{Stab}_S(R_x^U)$, $t \in \mathrm{Stab}_S(L_x^U)$ such that $y = sxt$. It follows that $sx\mathscr{R}^U x$ and $xt\mathscr{L}^U x$, and so, by Corollary 3.10, $sx\mathscr{L}^S x$ and $xt\mathscr{R}^S x$. Thus $sx \in L_x^S \cap R_x^U$, and so, from Proposition 3.12(c), $(sxx')\nu_x \in {}_xS$ and $((sxx')\nu_x)\Psi = (x'sxx'x)\mu_x = (x'sx)\mu_x \in ({}_xS)\Psi$. If $c \in \mathcal{C}$ is such that $(c)\mu_x$ is the representative of the left coset of $S_x \cap ({}_xS)\Psi$ containing $(x'sx)\mu_x$, then $(x'sxg)\mu_x = (x'sx)\mu_x \cdot (g)\mu_x = (c)\mu_x$ for some $g \in \mathrm{Stab}_S(L_x^U)$ such that $(g)\mu_x \in S_x \cap ({}_xS)\Psi$. Thus, by Lemma 3.8, $xx'sxg = xc$. But $sx \in R_x^U$ implies that $xx'sx = sx$ and so $sxg = xc$. Since $xg\mathscr{L}^U x$, it follows from Corollary 3.10(a) that $xg\mathscr{R}^S x$ and so $sxg\mathscr{R}^S sx$. But $xt\mathscr{R}^S x$ and so $y = sxt\mathscr{R}^S sx\mathscr{R}^S sxg = xc$.

If $y\mathscr{D}^S x$ is arbitrary, then $(y)\rho \sim (x)\rho$, by Proposition 3.22, and so there exists $i \in \{1, \ldots, m\}$ such that $(y)\rho = u_i \cdot (x)\rho$. By Proposition 2.3(a), there exists $\overline{u}_i \in S^1$ such that $u_i\overline{u_i}y = y$ and $\overline{u_i} \cdot (y)\rho = (x)\rho$.

Again by Proposition 3.22, $(x)\lambda \sim (y)\lambda$ and so there exists $v \in S^1$ such that $(y)\lambda \cdot v = (x)\lambda$. It follows that $y\mathscr{D}^S\overline{u}_iyv$ by Lemma 3.6(c). From Lemma 3.6(a), since $(y)\lambda \sim (x)\lambda = (yv)\lambda$, it follows that $y\mathscr{R}^Syv$ and so $(yv)\rho = (y)\rho$. Thus $(\overline{u}_iyv)\rho = \overline{u}_i \cdot (yv)\rho = \overline{u}_i \cdot (y)\rho = (x)\rho$ and so $\overline{u}_iyv\mathscr{R}^Ux$. Dually, from Lemma 3.6(b), $\overline{u}_iyv\mathscr{L}^Ux$ and so $\overline{u}_iyv \in D_x^S \cap H_x^U$. Hence there exists $c \in \mathcal{C}$ such that $\overline{u}_iyv\mathscr{R}^Sxc$ and so $y\mathscr{R}^Syv = u_i\overline{u}_iyv\mathscr{R}^Su_ixc$, as required.

Suppose there exist $i, j \in \{1, 2, \ldots, m\}$ and $c, d \in \mathcal{C}$ such that $u_ixc\mathscr{R}^Su_jxd$. Then, since $xc\mathscr{R}^Ux\mathscr{R}^Uxd$ (from the first paragraph), it follows that $(xc)\rho = (x)\rho = (xd)\rho$. Thus

$$u_i \cdot (x)\rho = u_i \cdot (xc)\rho = (u_ixc)\rho = (u_jxd)\rho = u_j \cdot (xd)\rho = u_j \cdot (x)\rho$$

and, by the minimality of $\{u_1, \ldots, u_m\}$, it follows that $u_i = u_j$ and $i = j$. By the analogue of Proposition 2.3(a), there exists $\overline{u_i}$ such that $\overline{u_i}u_ixc = xc$ and $\overline{u_i}u_ixd = xd$. Hence since $u_ixc\mathscr{R}^Su_jxd$ and $\mathscr{R}^S$ is a left congruence, it follows that $xc\mathscr{R}^Sxd$. If $xc = xd$, then, by Lemma 3.8, $(c)\mu_x = (d)\mu_x$, and by the minimality of $\mathcal{C}$, $c = d$. Suppose that $xc \neq xd$. Then there exists $y \in S$ such that $xcy = xd$. We showed above that $xc\mathscr{L}^Sx\mathscr{L}^Sxd$, and so $x\mathscr{L}^Sxd = xcy\mathscr{L}^Sxy$, and, in particular, $y \in \operatorname{Stab}_S(L_x^U)$. From Lemma 3.8 applied to $xcy = xd$, we deduce that $(c)\mu_x(y)\mu_x = (cy)\mu_x = (d)\mu_x$. This implies that $((c)\mu_x)^{-1}(d)\mu_x = (y)\mu_x \in S_x$. Therefore $(c)\mu_x$ and $(d)\mu_x$ are representatives of the same left coset of $S_x \cap ({}_xS)\Psi$ in $({}_xS)\Psi$, and again by the minimality of $\mathcal{C}$, $c = d$. □

Next, we give an analogue of Proposition 3.23 for $\mathscr{L}^S$- and $\mathscr{H}^S$-class representatives.

Proposition 3.24. *Suppose that $x \in S$ and there is $x' \in U$ with $xx'x = x$ and that:*

(a) *$\mathcal{C}$ is a minimal subset of $\operatorname{Stab}_S(L_x^U)$ such that $\{(c)\mu_x : c \in \mathcal{C}\}$ is a right transversal of $S_x \cap ({}_xS)\Psi$ in S_x where $\Psi : {}_xS \longrightarrow U_x$, defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$, is the embedding from Proposition 3.13(a);*

(b) *$\{v_1, \ldots, v_m\}$ is a minimal subset of S^1 such that $\{(x)\rho \cdot v_i : 1 \leq i \leq m\}$ equals the s.c.c. of $(x)\lambda$ under the right action of S on $(S)\lambda$.*

Then $\{xcv_i : c \in \mathcal{C},\ 1 \leq i \leq m\}$ is a minimal set of $\mathscr{H}^S$-class representatives of R_x^S, and hence a minimal set of $\mathscr{L}^S$-class representatives for D_x^S.

Proof. It follows from part (b) and Lemma 3.6(a) that $xcv_i\mathscr{R}^Sxc\mathscr{R}^Sx$, and so it suffices to show that $\{xcv_i : c \in \mathcal{C},\ 1 \leq i \leq m\}$ is a set of $\mathscr{L}^S$-class representatives for D_x^S. The proof is somewhat similar to that of Proposition 3.23, and so we will omit some details.

If $y \in D_x^S \cap H_x^U$, then we will show that there is $c \in \mathcal{C}$ such that $y\mathscr{L}^Sxc$. By Proposition 3.21, there exist $s \in \operatorname{Stab}_S(R_x^U)$ and $t \in \operatorname{Stab}_S(L_x^U)$ such that $y = sxt$. As in the proof of Proposition 3.23, it follows that $sx\mathscr{L}^Sx$ and $xt\mathscr{R}^Sx$. Thus $xt \in L_x^U \cap R_x^S$ and so $(x'xt)\mu_x \in S_x$. If $c \in \mathcal{C}$ is such that $(c)\mu_x$ is the representative of the right coset containing $(x'xt)\mu_x$, then there exists $g \in \operatorname{Stab}_S(R_x^U)$ such that $(x'gx)\mu_x \in S_x \cap ({}_xS)\Psi$ and $(x'gxt)\mu_x = (x'gxx'xt)\mu_x = (c)\mu_x$. Hence, by Lemma 3.8, $xx'gxt = xc$. But $xt\mathscr{R}^Sx$ and $g \in \operatorname{Stab}_S(R_x^U)$ and so $(gxt)\rho = g \cdot (xt)\rho = g \cdot (x)\rho = (x)\rho$, which implies that $gxt\mathscr{R}^Ux$. Since xx' is a left identity for R_x^U, $xx'gxt = gxt$, and so $gxt = xc$. Since $gx\mathscr{R}^Ux$, from Corollary 3.10(b), $gx\mathscr{L}^Sx$ and so $xc = gxt\mathscr{L}^Sxt$. Therefore $y = sxt\mathscr{L}^Sxt\mathscr{L}^Sxc$, as required.

The proof that an arbitrary $y \in D_x^S$ is $\mathscr{L}^S$-related to xcv_i for some i and that $(xcv_i, xdv_j) \notin \mathscr{L}^S$ if $i \neq j$ or $c \neq d$, is directly analogous to the final part of the proof of Proposition 3.23, and so we omit it. □

4 Specific classes of semigroups

In this section, we show how the results in Section 3 can be efficiently applied to transformation, partial permutation, matrix, and partition semigroups; and also to subsemigroups of finite regular Rees 0-matrix semigroups. More precisely, suppose that U is any of the full transformation monoid, the symmetric inverse monoid, the general linear monoid over any finite field, the partition monoid, or a finite regular Rees 0-matrix semigroup (the definitions of these semigroups can be found below) and that S is any subsemigroup of U. Then, as described at the start of Section 3, we will show that there exist homomorphisms λ and ρ of the actions of S on U by right and left multiplication whose kernels are $\mathscr{L}^U$ and $\mathscr{R}^U$, respectively, and where it is comparatively easy to compute with the actions of S on $(S)\lambda$ and $(S)\rho$. For such subsemigroups S, we also show how to obtain faithful representations of relatively small degrees of the stabilisers of $\mathscr{L}$- and $\mathscr{R}$-classes under the action of S.

4.1 Transformation semigroups

Let $n \in \mathbb{N}$ and write $\mathbf{n} = \{1, \ldots, n\}$. As already stated, a *transformation* of $\mathbf{n}$ is a function from $\mathbf{n}$ to itself, and the *full transformation monoid of degree* n, denoted T_n, is the monoid of all transformations on $\mathbf{n}$ under composition. We refer to subsemigroups of T_n as *transformation semigroups of degree* n. It is well-known that the full transformation monoid is regular; see [17, Exercise 2.6.15]. Hence, it is possible to apply the results from Section 3 to any transformation semigroup S.

Let $f \in T_n$ be arbitrary. Then the *image* of f is defined to be

$$\operatorname{im}(f) = \{(i)f : i \in \mathbf{n}\} \subseteq \mathbf{n}$$

and the *kernel* of f is defined by

$$\ker(f) = \{(i, j) : (i)f = (j)f\} \subseteq \mathbf{n} \times \mathbf{n}.$$

The kernel of a transformation is an equivalence relation, and every equivalence relation on $\mathbf{n}$ is the kernel of some transformation on $\mathbf{n}$. We will denote by $\mathcal{K}$ the set of all equivalence relations on $\mathbf{n}$. The *kernel classes* of a transformation $f \in T_n$, are just the equivalence classes of the equivalence relation $\ker(f)$.

The following well-known result characterises the Green's relations on the full transformation monoid.

Proposition 4.1 (Exercise 2.6.16 in [17].)**.** *Let $n \in \mathbb{N}$ and let $f, g \in T_n$. Then the following hold:*

(a) $f \mathscr{L}^{T_n} g$ *if and only if* $\operatorname{im}(f) = \operatorname{im}(g)$*;*

(b) $f \mathscr{R}^{T_n} g$ *if and only if* $\ker(f) = \ker(g)$*;*

(c) $f \mathscr{D}^{T_n} g$ *if and only if* $|\operatorname{im}(f)| = |\operatorname{im}(g)|$.

Proposition 4.2. *Let S be an arbitrary transformation semigroup of degree $n \in \mathbb{N}$. Then:*

(a) $\lambda : T_n \longrightarrow \mathcal{P}(\mathbf{n})$ *defined by* $(x)\lambda = \operatorname{im}(x)$ *is a homomorphism of the actions of S on T_n by right multiplication, and the natural action on $\mathcal{P}(\mathbf{n})$ and* $\ker(\lambda) = \mathscr{L}^{T_n}$*;*

(b) *if L is any $\mathscr{L}$-class of T_n, then S_L acts faithfully on* $\operatorname{im}(x)$ *for each $x \in L$;*

(c) $\rho : T_n \longrightarrow \mathcal{K}$ *defined by* $(x)\rho = \ker(x)$ *is a homomorphism of the actions of S on T_n by left multiplication, and the left action of S on $\mathcal{K}$ defined by*

$$x \cdot K = \ker(xy) \qquad \textit{where } y \in T_n,\ \ker(y) = K$$

and $\ker(\rho) = \mathscr{R}^U$*;*

(d) *if R is any $\mathscr{R}$-class of T_n, then ${}_RS$ acts faithfully on the set of kernel classes of* $\ker(x)$ *for each $x \in R$.*

Proof. We will only prove parts (a) and (b); the proofs of parts (c) and (d) are analogous.

(a). It follows from Proposition 4.1 that $\ker(\lambda) = \mathscr{L}^{T_n}$. If $x \in T_n$ and $s \in S$ are arbitrary, then $(xs)\lambda = \operatorname{im}(xs) = \operatorname{im}(x) \cdot s = (x)\lambda \cdot s$ and so λ is a homomorphism of the actions in part (a).

(b). Let $x \in L$ and let $\zeta : S_L \longrightarrow S_{\operatorname{im}(x)}$ be defined by $(s|_L)\zeta = s|_{\operatorname{im}(x)}$ where the action of $s|_{\operatorname{im}(x)}$ on $\operatorname{im}(x)$ (on the right) is defined by: $i \cdot (s|_{\operatorname{im}(x)}) = (i)s$, for all $i \in \operatorname{im}(x)$. Let $s \in \operatorname{Stab}_S(L)$ be arbitrary. Then $xs \in L$ and so $\operatorname{im}(xs) = \operatorname{im}(x)$, and, in particular, s acts on $\operatorname{im}(x)$. Thus ζ is well-defined. It is routine to verify that ζ is a homomorphism. From the definition of ζ, $s, t \in S$ have the same action on $\operatorname{im}(x)$ if and only if $xs = xt$. But, by Lemma 3.8, $xs = xt$ if and only if $s|_L = t|_L$ and the action of S_L on $\operatorname{im}(x)$ is faithful. ∎

4.2 Partial permutation semigroups and inverse semigroups

A *partial permutation* on $\mathbf{n} = \{1, \ldots, n\}$ is an injective function from a subset of $\mathbf{n}$ to another subset of equal cardinality. The *symmetric inverse monoid of degree* n, denoted I_n, is the monoid of all partial permutations on $\mathbf{n}$ under composition (as binary relations). We refer to subsemigroups of I_n as *partial permutation semigroups of degree* n. A semigroup U is called *inverse* if for every $x \in U$ there exists a unique $y \in U$ such that $xyx = x$ and $yxy = y$. Every inverse semigroup is isomorphic to an inverse subsemigroup of a symmetric inverse monoid by the Vagner-Preston Theorem; see [17, Theorem 5.1.7]. Since every inverse semigroup is regular, we may apply the results from Section 3 to arbitrary subsemigroups of the symmetric inverse monoid. We will give an analogue of Proposition 4.2 for subsemigroups of any symmetric inverse monoid over a finite set, for which we require a description of the Green's relations in I_n.

Let $f \in I_n$ be arbitrary. Then the *domain* of f is defined to be

$$\operatorname{dom}(f) = \{i \in \mathbf{n} : (i)f \text{ is defined}\} \subseteq \mathbf{n}$$

and the *image* of f is

$$\operatorname{im}(f) = \{(i)f : i \in \operatorname{dom}(f)\} \subseteq \mathbf{n}.$$

The *inverse* of f is the unique partial permutation f^{-1} with the property that $ff^{-1}f = f$ and $f^{-1}ff^{-1} = f^{-1}$; note that f^{-1} coincides with the usual inverse mapping $\operatorname{im}(f) \longrightarrow \operatorname{dom}(f)$.

Proposition 4.3 (Exercise 5.11.2 in [17].). *Let $n \in \mathbb{N}$ and let $f, g \in I_n$ be arbitrary. Then the following hold:*

(a) *$f \mathscr{L}^{I_n} g$ if and only if $\operatorname{im}(f) = \operatorname{im}(g)$;*

(b) *$f \mathscr{R}^{I_n} g$ if and only if $\operatorname{dom}(f) = \operatorname{dom}(g)$;*

(c) *$f \mathscr{D}^{I_n} g$ if and only if $|\operatorname{im}(f)| = |\operatorname{im}(g)|$.*

Proposition 4.4. *Let S be an arbitrary partial permutation semigroup of degree $n \in \mathbb{N}$. Then:*

(a) *$\lambda : I_n \longrightarrow \mathcal{P}(\mathbf{n})$ defined by $(x)\lambda = \operatorname{im}(x)$ is a homomorphism of the actions of S on I_n by right multiplication, and the right action on $\mathcal{P}(\mathbf{n})$ defined by*

$$A \cdot x = \{(a)x : a \in A \cap \operatorname{dom}(x)\} \qquad \text{for } A \in \mathcal{P}(\mathbf{n}) \text{ and } x \in I_n$$

and $\ker(\lambda) = \mathscr{L}^{I_n}$;

(b) *if L is any $\mathscr{L}$-class of I_n, then $(I_n)_L$ acts faithfully on the right of $\operatorname{im}(x)$ for each $x \in L$;*

(c) *$\rho : I_n \longrightarrow \mathcal{P}(\mathbf{n})$ defined by $(x)\rho = \operatorname{dom}(x)$ is a homomorphism of the actions of S on I_n by left multiplication, and the left action on $\mathcal{P}(\mathbf{n})$ defined by*

$$x \cdot A = \{(a)x^{-1} : a \in A \cap \operatorname{im}(x)\} \qquad \text{for } A \in \mathcal{P}(\mathbf{n}) \text{ and } x \in I_n;$$

and $\ker(\rho) = \mathscr{R}^{I_n}$;

(d) *if R is any $\mathscr{R}$-class of I_n, then ${}_RS$ acts faithfully on the left of $\operatorname{dom}(x)$ for each $x \in R$.*

Proof. The proof of this proposition is very similar to that of Proposition 4.2 and is omitted. □

4.3 Matrix semigroups

Let R be a finite field, let $n \in \mathbb{N}$, and let $M_n(R)$ denote the monoid of $n \times n$ matrices with entries in R (under the usual matrix multiplication). The monoid $M_n(R)$ is called a *general linear monoid*. In this paper, a *matrix semigroup* is a subsemigroup of some general linear monoid. It is well-known that $M_n(R)$ is a regular semigroup [32, Lemma 2.1].

If $\alpha \in M_n(R)$ is arbitrary, then denote by $r(\alpha)$ the *row space* of α (i.e. the subspace of the n-dimensional vector space over R spanned by the rows of α). We denote the *dimension* of $r(\alpha)$ by $\dim(r(\alpha))$. The notion of a *column space* and its dimension are defined dually. We denote the column space of $\alpha \in M_n(R)$ by $c(\alpha)$.

Proposition 4.5 (Lemma 2.1 in [32])**.** *Let R be a finite field, let $n \in \mathbb{N}$, and let $\alpha, \beta \in M_n(R)$ be arbitrary. Then the following hold:*

(a) *$\alpha \mathscr{L}^{M_n(R)} \beta$ if and only if $r(\alpha) = r(\beta)$;*

(b) *$\alpha \mathscr{R}^{M_n(R)} \beta$ if and only if $c(\alpha) = c(\beta)$;*

(c) *$\alpha \mathscr{D}^{M_n(R)} \beta$ if and only if $\dim(r(\alpha)) = \dim(r(\beta))$.*

Proposition 4.6. *Let R be a finite field, let $n \in \mathbb{N}$, and let S be an arbitrary subsemigroup of the general linear monoid $M_n(R)$. Then the following hold:*

(a) *if Ω denotes the collection of subspaces of R^n as row vectors, then $\lambda : M_n(R) \longrightarrow \Omega$ defined by $(\alpha)\lambda = r(\alpha)$ is a homomorphism of the actions of S on $M_n(R)$ by right multiplication, and the action on Ω by right multiplication, and $\ker(\lambda) = \mathscr{L}^{M_n(R)}$;*

(b) *if L is any $\mathscr{L}$-class of $M_n(R)$, then S_L acts faithfully on $r(\alpha)$ for each $\alpha \in L$;*

(c) *if Ω denotes the collection of subspaces of R^n as column vectors, then $\rho : M_n(R) \longrightarrow \Omega$ defined by $(\alpha)\rho = c(\alpha)$ is a homomorphism of the actions of S on $M_n(R)$ by left multiplication, and the action on Ω by left multiplication, and $\ker(\rho) = \mathscr{R}^{M_n(R)}$;*

(d) *if R is any $\mathscr{R}$-class of $M_n(R)$, then ${}_RS$ acts faithfully on $c(\alpha)$ for each $\alpha \in R$.*

Proof. We will only prove (a) and (b); the proofs of parts (c) and (d) follow by analogous arguments. We will write L_α to mean the $\mathscr{L}$-class of $\alpha \in M_n(R)$ in $M_n(R)$ throughout this proof.

(a). It follows from Proposition 4.5(a) that $(\alpha)\lambda = (\beta)\lambda$ if and only if $\alpha \mathscr{L}^{M_n(R)} \beta$, and so $\ker(\lambda) = \mathscr{L}^{M_n(R)}$. We also have

$$(\alpha\beta)\lambda = r(\alpha\beta) = r(\alpha) \cdot \beta = (\alpha)\lambda \cdot \beta$$

for all $\alpha \in M_n(R)$ and $\beta \in S$, and so λ is a homomorphism of actions.

(b). Let L be any $\mathscr{L}$-class in $M_n(R)$, let $\alpha \in L$ and let $\beta, \gamma \in \mathrm{Stab}_S(L)$. Then $\alpha\beta \in L$ and so β acts on $r(\alpha)$ by right multiplication. By Lemma 3.8, it follows that β and γ have equal action on $r(\alpha)$ if and only if $\alpha\beta = \alpha\gamma$ if and only if $\alpha|_L = \beta|_L$. $\square$

4.4 Subsemigroups of a Rees 0-matrix semigroup

In this section, we describe how the results from Section 3 can be applied to subsemigroups of a Rees 0-matrix semigroup. We start by recalling the relevant definitions.

Let T be a semigroup, let 0 be an element not in T, let I and J be sets, and let $P = (p_{j,i})_{j \in J, i \in I}$ be a $|J| \times |I|$ matrix with entries from $T \cup \{0\}$. Then the *Rees 0-matrix semigroup* $\mathcal{M}^0[T; I, J; P]$ is the set $(I \times T \times J) \cup \{0\}$ with multiplication defined by

$$0x = x0 = 0 \text{ for all } x \in \mathcal{M}^0[T; I, J; P] \qquad \text{and} \qquad (i, g, j)(k, h, l) = \begin{cases} (i, gp_{j,k}h, l) & \text{if } p_{j,k} \neq 0 \\ 0 & \text{if } p_{j,k} = 0. \end{cases}$$

A semigroup U with a zero element 0 is 0-*simple* if U and $\{0\}$ are its only ideals.

Theorem 4.7 (Theorem 3.2.3 in [17] or Theorem A.4.15 in [36])**.** *A finite semigroup U is 0-simple if and only if it is isomorphic to a Rees 0-matrix semigroup $\mathcal{M}^0[G; I, J; P]$, where G is a group, and P is regular, in the sense that every row and every column contains at least one non-zero entry.*

Green's relations of a regular Rees 0-matrix semigroup are described in the following proposition.

Proposition 4.8. *Let $U = \mathcal{M}^0[G; I, J; P]$ be a finite Rees 0-matrix semigroup where G is a group and P is regular. Then the following hold for all $x, y \in U$:*

(a) *$x \mathscr{L}^U y$ if and only if $x, y \in I \times G \times \{j\}$ for some $j \in J$ or $x = y = 0$.*

(b) *$x \mathscr{R}^U y$ if and only if $x, y \in \{i\} \times G \times J$ for some $i \in I$ or $x = y = 0$;*

Obviously, we do not require any theory beyond that given above to compute with Rees 0-matrix semigroups, since their size, elements, and in the case that they are regular, their Green's structure and maximal subgroups too, are part of their definition. However, it might be that we would like to compute with a proper subsemigroup of a Rees 0-matrix semigroup. Several computational problems for arbitrary finite semigroups can be reduced, in part, to problems for associated Rees 0-matrix semigroups (the principal factors of certain $\mathscr{D}$-classes). For example, this is the case for finding the automorphism group [2], minimal (idempotent) generating sets [9, 16], or the maximal subsemigroups of a finite semigroup. In the latter example, we may wish to determine the structure of the maximal subsemigroups, which are not necessarily Rees 0-matrix semigroups themselves. In the absence of a method to find a convenient representation of a subsemigroup of a Rees 0-matrix semigroup, as, for example, a transformation semigroup, we would have to compute directly with the subsemigroup.

Proposition 4.9. *Let S be an arbitrary subsemigroup of a finite regular Rees 0-matrix semigroup $U = \mathcal{M}^0[G; I, J; P]$ over a permutation group G acting faithfully on $\mathbf{n}$ for some $n \in \mathbb{N}$. Then:*

(a) *$\lambda : U \longrightarrow J \cup \{0\}$ defined by $(i, g, j)\lambda = j$ and $(0)\lambda = 0$ is a homomorphism of the actions of S on U by right multiplication, and the right action of S on $J \cup \{0\}$ defined by*

$$0 \cdot (i, g, j) = 0 \cdot 0 = 0 = k \cdot 0 \quad \text{and} \quad k \cdot (i, g, j) = \begin{cases} j & \text{if } p_{k,i} \neq 0 \\ 0 & \text{if } p_{k,i} = 0 \end{cases}$$

for all $k \in J$, and $\ker(\lambda) = \mathscr{L}^U$;

(b) *if L is any non-zero $\mathscr{L}$-class of U, then the action of S_L on $\mathbf{n}$ defined by*

$$m \cdot (i, g, j)|_L = (m)p_{j,i}g \quad \text{for all} \quad m \in \mathbf{n}$$

is faithful;

(c) *$\rho : U \longrightarrow I \cup \{0\}$ defined by $(i, g, j)\rho = i$ and $(0)\rho = 0$ is a homomorphism of the actions of S on U by left multiplication, and the left action of S on $I \cup \{0\}$ defined by*

$$(i, g, j) \cdot 0 = 0 \cdot 0 = 0 = 0 \cdot k \quad \text{and} \quad (i, g, j) \cdot k = \begin{cases} i & \text{if } p_{j,k} \neq 0 \\ 0 & \text{if } p_{j,k} = 0 \end{cases}$$

for all $k \in I$, and $\ker(\rho) = \mathscr{R}^U$;

(d) *if R is any non-zero $\mathscr{R}$-class of U, then the action of ${}_RS$ on $\mathbf{n}$ defined by*

$$_R|(i, g, j) \cdot m = (m)g^{-1}p_{j,i}^{-1} \quad \text{for all} \quad m \in \mathbf{n}$$

is faithful.

Proof. We only prove parts (a) and (b); parts (c) and (d) follow by analogous arguments.

(a). It follows by Proposition 4.8(a) that $(x)\lambda = (y)\lambda$ if and only if $x\mathscr{L}^U y$, for each $x, y \in U$, and so the kernel of λ is $\mathscr{L}^U$. We will show that λ is a homomorphism of actions.

Let $x \in U$ and $s \in S$ be arbitrary. We must show that $(xs)\lambda = (x)\lambda \cdot s$. If $x = 0$ or $s = 0$, then $(xs)\lambda = (0)\lambda = 0 = (x)\lambda \cdot s$. Suppose that $x = (i, g, j) \in U \setminus \{0\}$ and $s = (k, h, l) \in S \setminus \{0\}$. If $p_{j,k} = 0$, then $xs = 0$ and so

$$(xs)\lambda = (0)\lambda = 0 = j \cdot (k, h, l) = (x)\lambda \cdot s.$$

If $p_{j,k} \neq 0$, then

$$(xs)\lambda = l = j \cdot (k, h, l) = (x)\lambda \cdot s.$$

(b). Let $x = (i, g, j) \in U \setminus \{0\}$ and let $L = L^U_x = \{(i', g', j) : i' \in I,\ g' \in G\}$. If $(k, h, l) \in \mathrm{Stab}_S(L)$ is arbitrary, then, since $L \cdot (k, h, l) = L$, it follows that $p_{j,k} \neq 0$ and $l = j$. It follows that we may define a mapping $\zeta : S_L \longrightarrow G$ by $((k, h, l)|_L)\zeta = p_{j,k}h$.

If $(k_1,h_1,j),(k_2,h_2,j)\in \mathrm{Stab}_S(L)$, then, by Lemma 3.8, it follows that

$$\begin{array}{lll} (k_1,h_1,j)|_L=(k_2,h_2,j)|_L & \text{if and only if} & (i,g,j)(k_1,h_1,j)=(i,g,j)(k_2,h_2,j)\\ & \text{if and only if} & (i,gp_{j,k_1}h_1,j)=(i,gp_{j,k_2}h_2,j)\\ & \text{if and only if} & p_{j,k_1}h_1=p_{j,k_2}h_2. \end{array}$$

Hence, ζ is well-defined and injective.

To show that ζ is a homomorphism, suppose that $(k_1,h_1,j),(k_2,h_2,j)\in \mathrm{Stab}_S(L)$. Then

$$\begin{array}{rcl} ((k_1,h_1,j)|_L(k_2,h_2,j)|_L)\zeta &=& (((k_1,h_1,j)(k_2,h_2,j))|_L)\zeta\\ &=& ((k_1,h_1p_{j,k_2}h_2,j)|_L)\zeta\\ &=& p_{j,k_1}h_1p_{j,k_2}h_2\\ &=& ((k_1,h_1,j)|_L)\zeta\cdot((k_2,h_2,j)|_L)\zeta, \end{array}$$

as required. □

4.5 Partition monoids

Let $n\in\mathbb{N}$, let $\mathbf{n}=\{1,\ldots,n\}$, and let $-\mathbf{n}=\{-1,\ldots,-n\}$. A *partition* of $\mathbf{n}\cup-\mathbf{n}$ is a set of pairwise disjoint non-empty subsets of $\mathbf{n}\cup-\mathbf{n}$ (called *blocks*) whose union is $\mathbf{n}\cup-\mathbf{n}$. If $i,j\in\mathbf{n}\cup-\mathbf{n}$ belong to the same block of a partition x, then we write $(i,j)\in x$.

If x and y are partitions of $\mathbf{n}\cup-\mathbf{n}$, then we define the product xy of x and y to be the partition where for $i,j\in\mathbf{n}$

(i) $(i,j)\in xy$ if and only if $(i,j)\in x$ or there exist $a_1,\ldots,a_{2r}\in\mathbf{n}$, for some $r\geq 1$, such that

$$(i,-a_1)\in x,\quad (a_1,a_2)\in y,\quad (-a_2,-a_3)\in x,\quad \ldots,\quad (a_{2r-1},a_{2r})\in y,\quad (-a_{2r},j)\in x$$

(ii) $(i,-j)\in xy$ if and only if there exist $a_1,\ldots,a_{2r-1}\in\mathbf{n}$, for some $r\geq 1$, such that

$$(i,-a_1)\in x,\quad (a_1,a_2)\in y,\quad (-a_2,-a_3)\in x,\quad \ldots,\quad (-a_{2r-2},-a_{2r-1})\in x,\quad (a_{2r-1},-j)\in y$$

(iii) $(-i,-j)\in xy$ if and only if $(-i,-j)\in y$ or there exist $a_1,\ldots,a_{2r}\in\mathbf{n}$, for some $r\geq 1$, such that

$$(-i,a_1)\in y,\quad (-a_1,-a_2)\in x,\quad (a_2,a_3)\in y,\quad \ldots,\quad (-a_{2r-1},-a_{2r})\in x,\quad (a_{2r},-j)\in y$$

for $i,j\in\mathbf{n}$.

This product can be shown to be associative, and so the collection of partitions of $\mathbf{n}\cup-\mathbf{n}$ is a monoid; the identity element is the partition $\{\{i,-i\}:i\in\mathbf{n}\}$. This monoid is called the *partition monoid* and is denoted P_n.

It can be useful to represent a partition as a graph with vertices $\mathbf{n}\cup-\mathbf{n}$ and the minimum number of edges so that the connected components of the graph correspond to the blocks of the partition. Of course, such a representation is not unique in general. An example is given in Figure 1 for the partitions:

$$\begin{array}{rcl} x &=& \big\{\{1,-1\},\{2\},\{3\},\{4,-3\},\{5,6,-5,-6\},\{-2,-4\}\big\}\\ y &=& \big\{\{1,4,-1,-2,-6\},\{2,3,5,-4\},\{6,-3\},\{-5\}\big\} \end{array}$$

and the product

$$xy \;=\; \big\{\{1,4,5,6,-1,-2,-3,-4,-6\},\{2\},\{3\},\{-5\}\big\}$$

is shown in Figure 2.

A block of a partition containing elements of both $\mathbf{n}$ and **-n** is called a *transverse block*. If $x\in P_n$, then we define x^* to be the partition obtained from x by replacing i by $-i$ and $-i$ by i in every block of x for all $i\in\mathbf{n}$. It is routine to verify that if $x,y\in P_n$, then

$$(x^*)^*=x,\quad xx^*x=x,\quad x^*xx^*=x^*,\quad (xy)^*=y^*x^*.$$

In this way, the partition monoid is a *regular $*$-semigroup* in the sense of [31].

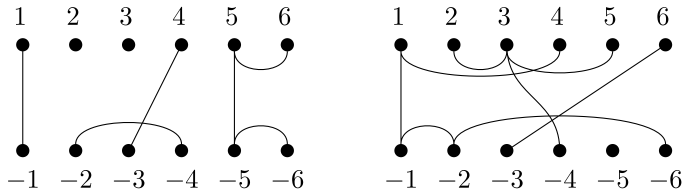

Figure 1: Graphical representations of the partitions $x, y \in P_6$.

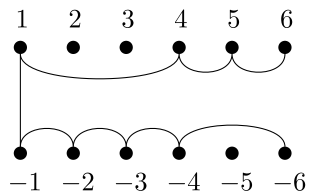

Figure 2: A graphical representation of the product $xy \in P_6$.

If $x \in P_n$ is arbitrary, then xx^* and x^*x are idempotents; called the *projections* of x. We will write

$$\mathrm{Proj}(P_n) = \{xx^* : x \in P_n\} = \{x^*x : x \in P_n\}.$$

If B is a transverse block of xx^* (or x^*x), then $i \in B$ if and only if $-i \in B$. If B is a non-transverse block of xx^*, then $-B = \{-b : b \in B\}$ is also a block of xx^*.

Proposition 4.10 (cf. [11, 44])**.** *Let $n \in \mathbb{N}$ and let $x, y \in P_n$. Then the following hold:*

(a) *$x \mathscr{L}^{P_n} y$ if and only if $x^*x = y^*y$;*

(b) *$x \mathscr{R}^{P_n} y$ if and only if $xx^* = yy^*$;*

(c) *$x \mathscr{D}^{P_n} y$ if and only if x and y have the same number of transverse blocks.*

The characterisation in Proposition 4.10 can be used to define representations of the actions mentioned above.

Proposition 4.11. *Let S be an arbitrary subsemigroup of P_n. Then:*

(a) *$\lambda : P_n \longrightarrow \mathrm{Proj}(P_n)$ defined by $(x)\lambda = x^*x$ is a homomorphism between the action of S on P_n by right multiplication and the right action of S on $\mathrm{Proj}(P_n)$ defined by*

$$x^*x \cdot y = (xy)^*xy = y^*x^*xy$$

and the kernel of λ is $\mathscr{L}^{P_n}$;

(b) *if L is any $\mathscr{L}$-class of P_n, then S_L acts faithfully on the transverse blocks of x^*x for each $x \in L$;*

(c) *$\rho : P_n \longrightarrow \mathrm{Proj}(P_n)$ defined by $(x)\rho = xx^*$ is a homomorphism between the action of S on P_n by left multiplication and the left action of S on $\mathrm{Proj}(P_n)$ defined by*

$$y \cdot xx^* = yx(yx)^* = yxx^*y^*$$

and the kernel of ρ is $\mathscr{R}^{P_n}$;

(d) *if R is any $\mathscr{R}$-class of P_n, then ${}_RS$ acts faithfully on the transverse blocks of xx^* for each $x \in R$.*

Proof. We just prove parts (a) and (b), since the other parts are dual.

(a). Let $x \in P_n$ and $s \in S$ be arbitrary. Then

$$(xs)\lambda = (xs)^*xs = s^*(x^*x)s = (x)\lambda \cdot s.$$

Together with Proposition 4.10, this completes the proof of (a).

(b). Let $x \in P_n$ be arbitrary and suppose that $y \in \mathrm{Stab}_S(L_x^{P_n})$. It follows that $x^*x = (x)\lambda = (xy)\lambda = (xy)^*xy = y^*x^*xy$. We denote the intersection of the transverse blocks of x^*x with $\mathbf{n}$ by $B_1, \dots, B_r$ and we define the binary relation

$$p_y = \{(i, j) \in \mathbf{r} \times \mathbf{r} : \exists k \in B_i,\ \exists l \in B_j,\ (k, -l) \in x^*xy\}.$$

We will show that p_y is a permutation, and that $\zeta : S_{L_x^{P_n}} \longrightarrow \mathrm{Sym}(\mathbf{r})$ defined by $(y|_{L_x^{P_n}})\zeta = p_y$ is a monomorphism.

Seeking a contradiction, assume that there exist $i, j, j' \in \mathbf{r}$ such that $j \neq j'$ and $(i, j), (i, j') \in p_y$. Then there exist $k, k' \in B_i$, $l \in B_j$, and $l' \in B_{j'}$ such that $(k, -l), (k', -l') \in x^*xy$. Since $k, k' \in B_i$, it follows that $(k, k') \in x^*x$ and so $(k, k') \in x^*xy$. Since x^*xy is an equivalence relation, it follows that $(-l, -l') \in x^*xy$, which implies that $(-l, -l') \in y^*x^*xy = x^*x$, and so $(l, l') \in x^*x$, a contradiction. Hence p_y is a function.

Since $\mathbf{n}$ is finite, to show that p_y is a permutation it suffices to show that it is surjective. Suppose that $i \in \mathbf{r}$ and $l \in B_i$ are arbitrary. Then $(l, -l) \in x^*x = y^*x^*xy$, and so by part (ii) of the definition of the multiplication of the partitions y^* and x^*xy, there exists $k \in \mathbf{n}$ such that $(k, -l) \in x^*xy$. In other words, k belongs to a transverse block of x^*xy, and hence to a transverse block of x^*x. Thus there exists $j \in \mathbf{r}$ such that $k \in B_j$ and so $(j)p_y = i$, as required.

Note that, since x^*xy is an equivalence relation and p_y is a permutation, the transverse blocks of x^*xy are of the form $B_i \times -B_{(i)p_y}$.

Next, we show that $\zeta : S_{L_x^{P_n}} \longrightarrow \mathrm{Sym}(\mathbf{r})$ defined by $(y|_{L_x^{P_n}})\zeta = p_y$ is a homomorphism. Let $y, z \in \mathrm{Stab}_S(L_x^{P_n})$. It suffices to prove that $p_y p_z = p_{yz}$. If $i \in \mathbf{r}$, then there exist $k \in B_i$, $l, l' \in B_{(i)p_y}$, and $m \in B_{((i)p_y)p_z}$ such that $(k, -l) \in x^*xy$ and $(l', -m) \in x^*xz$. Since $l, l' \in B_{(i)p_y}$ (a transverse block of x^*x), $(l, -l') \in x^*x$, and so $(k, -m) \in x^*xy \cdot x^*x \cdot x^*xz = x^*xyx^*xz$. Since $xy \mathscr{L}^{P_n} x$ and x^*x is a right identity in its $\mathscr{L}^{P_n}$-class, it follows that $xyx^*x = xy$ and so $x^*xyx^*xz = x^*xyz$. In particular, $(k, -m) \in x^*xyz$, and so $(i)p_{yz} = ((i)p_y)p_z$, as required.

It remains to prove that ζ is injective. Suppose that $y, z \in \mathrm{Stab}_S(L_x^{P_n})$ are such that $p_y = p_z$. It suffices, by Lemma 3.8, to show that $xy = xz$. We will prove that $x^*xy = x^*xz$ so that $xy = xx^*xy = xx^*xz = xz$. Since the transverse blocks of x^*xy are $B_i \times -B_{(i)p_y} = B_i \times -B_{(i)p_z}$ where $i \in \mathbf{r}$, it follows that the transverse blocks of x^*xy and x^*xz coincide. Suppose that $(k, l) \in x^*xy$ where neither k nor l belongs to a transverse block of x^*xy. It follows from the form of the transverse blocks of x^*xy that neither k nor l belongs to a transverse block of x^*x. There are two cases to consider: $k, l > 0$ and $k, l < 0$. In the first case, by part (i) of the definition of the mulitplication of x^*x and y, either $(k, l) \in x^*x$ or k and l belong to transverse blocks of x^*x. Since the latter is not the case, $(k, l) \in x^*x$ and so $(k, l) \in x^*xz$. In the second case, when $k, l < 0$, it follows from part (iii) of the definition of the multiplication of x^*x and y, that $(k, l) \in y$ and so $(k, l) \in y^*x^*xy = z^*x^*xz$. By part (iii) of the definition of the multiplication of z^* and x^*xz, either $(k, l) \in x^*xz$ or both k and l belong to transverse blocks of x^*xz. Since the transverse blocks of x^*xy and x^*xz coincide and contain neither k nor l, it is the case that $(k, l) \in x^*xz$. Thus $x^*xy \subseteq x^*xz$ and, by symmetry, $x^*xz \subseteq x^*xy$. Therefore $xy = xz$, as required. □

5 Algorithms

In this section, we outline some algorithms for computing with semigroups that utilise the results in Section 3. The algorithms described in this section are implemented in the GAP [15] package Semigroups [28] in their full generality, and can currently be applied to semigroups of transformations, partial permutations, partitions, and to subsemigroups of regular Rees 0-matrix semigroups over groups.

Throughout this section, we assume that U is a finite regular semigroup, that S is a subsemigroup of U generated by $X = \{x_1, \dots, x_m\} \subseteq U$ for some $m \in \mathbb{N}$. If U or S is not a monoid, then we can simply adjoin an identity, to obtain U^1 or S^1, perform whatever calculation we require in U^1 or S^1 and then return the answer for U. In other words, we may assume without loss of generality that U is a monoid and that S is a submonoid of U. We denote the identity of U by 1_U.

The purpose of the algorithms described in this section is to answer various questions about the structure of the semigroup $S = \langle x_1, \dots, x_m \rangle$ generated by $X = \{x_1, \dots, x_m\} \subseteq U$.

Apart from this introduction, this section has 6 subsections. In Subsection 5.1, we outline some basic operations that we must be able to compute in order to apply the algorithms later in this section. We then show how to perform these basic operations in the examples of semigroups of transformations, partial permutations, matrices, partitions, and in subsemigroups of a Rees 0-matrix semigroup in Subsection 5.2. In Subsection 5.3, we describe how to calculate the components of the action of a semigroup on a set, and how to use this to obtain the Schreier generators from Proposition 2.3(c). In Subsection 5.4, we describe a data structure for individual Green's classes of S and give algorithms showing how this data structure can be used to compute various properties of these classes. In Subsection 5.5, we describe algorithms that can be used to find global properties of S, such as its size, $\mathscr{R}$-classes, and so on. In Subsection 5.6, we give details of how some of the algorithms in Subsection 5.5 can be optimised when it is known *a priori* that S is a regular or inverse semigroup.

At several points in this section it is necessary to be able to determine the strongly connected components (s.c.c.) of a directed graph. This can be achieved using Tarjan's [43] or Gabow's [14] algorithms, for example; see also Sedgwick [38].

In the algorithms in this section, ":=" indicates that we are assigning a value (the right hand side of the expression) to a variable (the left hand side), while "=" denotes a comparison of variables. The symbol "$\leftarrow$" is the replacement operator, used to indicated that the value of the variable on the left hand side is replaced by the value on the right hand side.

5.1 Assumptions

As discussed at the start of Section 3, we suppose that we have a right action of S on a set $(U)\lambda$ and a homomorphism $\lambda : U \longrightarrow (U)\lambda$ of this action and the action of S on $\mathcal{P}(U)$ by right multiplication, where $\ker(\lambda) = \mathscr{L}^U$ (Definition 3.4). Furthermore, for every $x \in U$ we assume that we have a faithful representation ζ of the stabiliser $U_{L_x^U}$ and a function $\mu_x : \mathrm{Stab}_U(L_x^U) \longrightarrow (U_{L_x^U})\zeta$ defined by

$$(u)\mu_x = (u|_{L_x^U})\zeta \qquad \text{for all} \qquad u \in U;$$

see (3.7). We also assume that we have the left handed analogues $\rho : U \longrightarrow (U)\rho$ and $\nu_x : \mathrm{Stab}_U(R_x^U) \longrightarrow ({}_{R_x^U}U)\zeta'$ (where ζ' is any faithful representation of ${}_{R_x^U}U$) of λ and μ_x, respectively. Recall that we write

$$S_x = (\mathrm{Stab}_S(L_x^U))\mu_x \quad \text{and} \quad {}_xS = (\mathrm{Stab}_S(R_x^U))\nu_x.$$

In order to apply the algorithms described in this section, it is necessary that certain fundamental computations can be performed.

Assumptions. We assume that we can compute the following:

(I) the product xy;

(II) the value $(x)\lambda$;

(III) an element $\overline{s} \in U$ such that $xs\overline{s} = x$ whenever $(x)\lambda \sim (x)\lambda \cdot s = (xs)\lambda$;

(IV) the value $(x's)\mu_x$, for some choice of $x' \in U$ such that $xx'x = x$, whenever $(x)\lambda = (s)\lambda$ and $(x)\rho = (s)\rho$;

for all $x, y, s \in U$. We also require the facility to perform the analogous computations involving the functions ρ and the ν_x for all $x \in S$.

We will prove that $\overline{s} \in S$ in Assumption (III) exists. By the definition of λ, since $(x)\lambda \sim (xs)\lambda$, it follows that $L_x^U \sim L_{xs}^U$ under the action of S on $U/\mathscr{L}$ defined in (3.2). Hence, by Proposition 2.3(a), there exists $\overline{s} \in S$ such that

$$L_{xs}^U \cdot \overline{s} = L_x^U \quad \text{and} \quad (s\overline{s})|_{L_x^U} = \mathrm{id}_{L_x^U}$$

and so $xs\overline{s} = x$. Given the orbit graph of $(x)\lambda \cdot S$, it is possible to compute $\overline{s}$ by finding a path from $(xs)\lambda$ to $(x)\lambda$ and using Algorithm 2. However, this is often more expensive than computing $\overline{s} \in U$ directly from s. Some details of how to compute Assumptions (I) to (IV) in the special cases given in Section 4 can be found in the next section.

If $\overline{s} \in U$ satisfies Assumption (III) for some $x \in S$, then we note that $xs\overline{s} = x$ implies that $xs\overline{s}s = xs$ and so by Lemma 3.8:

$$(\overline{s}s)|_{L_{xs}^U} = \mathrm{id}_{L_{xs}^U}.$$

Note that if $s \in \mathrm{Stab}_S(R_x^U)$, then $(sx)\rho = (x)\rho$ and, by Proposition 3.12(b), $sx \in L_x^S \cap R_x^U$. In particular, $(sx)\lambda = (x)\lambda$, and so, by Assumption (IV), we can compute:

(V) the value $((s)\nu_x)\Psi = (x'sx)\mu_x$ whenever $s \in \mathrm{Stab}_S(R^U_x)$.

We will refer to this as Assumption (V), even though it is not an assumption.

It will follow from the comments in Subsection 5.2 that the algorithms described in this paper can be applied to any subsemigroup of the full transformation monoid, the symmetric inverse monoid, the partition monoid, the general linear monoid, or any subsemigroup of a regular Rees 0-matrix semigroup over a group. However, we would like to stress that the algorithms in this section apply to any subsemigroup of a finite regular semigroup. In the worst case, the functions $\lambda : U \longrightarrow U/\mathscr{L}^U$ and $\rho : U \longrightarrow U/\mathscr{R}^U$ defined by $(x)\lambda = L^U_x$ and $(x)\rho = R^U_x$, and the natural mappings $\mu_x : \mathrm{Stab}_S(L^U_x) \longrightarrow S_{L^U_x}$ and $\nu_x : \mathrm{Stab}_S(R^U_x) \longrightarrow {}_{R^U_x}S$ for every $x \in U$, fulfil the required conditions, although it might be that our algorithms are not very efficient in this case.

5.2 Computational prerequisites

In this section, we describe how to perform the computations required in Assumptions (I) to (IV) for semigroups of transformations, partial permutations, partitions, and matrices, and for subsemigroups of a Rees 0-matrix semigroup.

Transformations

A transformation x can be represented as a tuple $((1)x, \ldots, (n)x)$ where n is the degree of x.

(I) The composition xy of transformations x and y is represented by $((1)xy, \ldots, (n)xy)$, which can be computed by simple substitution in linear time $O(n)$;

(II) The value $(x)\lambda = \mathrm{im}(x)$ can be found by sorting and removing duplicates from $((1)x, \ldots, (n)x)$ with complexity $O(n \log(n))$;

(III) If x, s are such that $\mathrm{im}(x)$ and $\mathrm{im}(xs)$ have equal cardinality, then the transformation $\overline{s}$ defined by

$$(i)\overline{s} = \begin{cases} (j)x & \text{if } i = (j)xs \in \mathrm{im}(xs) \\ i & \text{if } i \notin \mathrm{im}(xs) \end{cases}$$

has the property that $xs\overline{s} = x$ (finding $\overline{s}$ has complexity $O(n)$);

(IV) If x and s are transformations such that $\ker(x) = \ker(s)$ and $\mathrm{im}(x) = \mathrm{im}(s)$ and x' is any transformation such that $xx'x = x$, then, from Proposition 4.2(b), $(x's)\mu_x$ is just the restriction $(x's)|_{\mathrm{im}(x)}$ of $x's$ to $\mathrm{im}(x)$, which can be determined in $|\mathrm{im}(x)|$ steps from x and s.

The analogous calculations can be made in terms of the kernel of a transformation, and the left actions of S; the details are omitted.

Partial permutations

A partial permutation x can be represented as a tuple $((1)x, \ldots, (m)x)$ where m is the largest value where x is defined, and $(i)x = 0$ if x is not defined at i. The values required under Assumptions (I), (II), and (IV) can be computed for partial permutations in the same way they were computed for transformations.

Assumption (III) is described below:

(III) if x, s are such that $\mathrm{im}(x)$ and $\mathrm{im}(xs)$ have equal cardinality, then the partial permutation $\overline{s} = s^{-1}$ has the property $xs\overline{s} = x$ (finding s^{-1} has complexity $O(n)$).

Matrices over finite fields

The values required in Assumptions (I) to (IV) can be found for matrix semigroups in a similar way as they are found for transformation and partial permutations, using elementary linear algebra; the details will appear in [29].

Rees 0-matrix semigroups

It should be clear from the definition of a Rees 0-matrix semigroup, and by Proposition 4.9, how to compute the values in Assumptions (I) and (II). Assumptions (III) and (IV) are described below:

(III) if $x = (a, b, i), s = (j, g, k) \in S$ are such that $(x)\lambda = i \sim (s)\lambda = k$, then $p_{i,j} \neq 0$. So, if $l \in I$ is such that $p_{k,l} \neq 0$, then $\overline{s} = (l, p_{k,l}^{-1}g^{-1}p_{i,j}^{-1}, i)$ has the property that $xs\overline{s} = x$;

(IV) if $x = (i, g, j) \in S$, then there exist $k \in I$ and $l \in J$ such that $p_{j,k}, p_{l,i} \neq 0$. One choice for $x' \in U$ is $(k, p_{j,k}^{-1}g^{-1}p_{l,i}^{-1}, l)$. So, if $h \in G$ is arbitrary and $s = (i, h, j)$, then $(x)\lambda = (s)\lambda$ and $(x)\rho = (s)\rho$. Hence the action of $(x's)\mu_x$ on $m \in \mathbf{n}$ (defined in Proposition 4.9(b)) is given by

$$m \cdot (x's)\mu_x = (m)g^{-1}h.$$

Partitions

A partition $x \in P_n$ can be represented as a $2n$-tuple where the first n entries correspond to the indices of the blocks containing $\{1, \ldots, n\}$ and entries $n+1$ to $2n$ correspond to the indices of the blocks containing $\{-1, \ldots, -n\}$. For example, the partition $x \in P_6$ shown in Figure 1 is represented by $(1, 2, 3, 4, 5, 5, 1, 6, 4, 6, 5, 5)$.

Given $x \in P_n$ represented as above, it is possible to compute $x^* \in P_n$ in $2n$ steps (linear complexity). This will be used in several of the assumptions.

(I) The composition xy of partitions x and y can be found using a variant of the classical Union-Find Algorithm (complexity $O(n^2)$);

(II) The value x^*x can be found using (I) (complexity $O(n^2)$);

(III) If $x, s \in P_n$ are such that $(x)\lambda = x^*x \sim (xs)\lambda = s^*x^*xs$, then x^*x and s^*x^*xs have equal number of transverse blocks. A partition $\overline{s}$ with the property that $xs\overline{s} = x$ can then be found using a variant of the Union-Find Algorithm (complexity $O(n^2)$);

(IV) If $x, s \in P_n$ are such that $(x)\lambda = (s)\lambda$ and $(x)\rho = (s)\rho$, then, by Proposition 4.11(b), $(x^*s)\mu_x$ is a permutation of the transverse blocks of x, which can be found in linear time (complexity $O(n)$).

In practice, it is possible to compute the values required by these assumptions with somewhat better complexity than that given above. However, this is also more complicated to describe and so we opted to describe the simpler methods.

5.3 Components of the action

Let S be an arbitrary monoid acting on a set Ω on the right, and let 1_S denote the identity of S. We start by describing a procedure for calculating $\alpha \cdot S = \{\alpha \cdot s : s \in S\}$ or $\Omega \cdot S = \{\alpha \cdot s : \alpha \in \Omega,\ s \in S\}$. These are essentially the same as the standard orbit algorithm for a group acting on a set (see, for example, [45, Section 4.1]), but without the assumption that S is a group. An analogous algorithm can be used for left actions. We will refer to $\alpha \cdot S$ as the *component of the action* under S of α.

In this subsection we present algorithms for computing: the components of an action of a semigroup S; elements of S that act on points in the component in a specified way; generators for the stabiliser of a set. Examples of how these algorithms can applied can be found in Section 6. The algorithms in this subsection and the next are somewhat similar to those described in [24].

Suppose that $X = \{x_1, \ldots, x_m\}$ is a generating set for a monoid S acting on the right on a set Ω. If $\alpha \in \Omega$ and $\alpha \cdot S = \{\beta_1 = \alpha, \beta_2, \ldots, \beta_n\}$, then the *orbit graph* of $\alpha \cdot S$ is just the directed graph with vertices $\{1, \ldots, n\}$ and an edge from i to $g_{i,j}$ labelled with j if $\beta_i \cdot x_j = \beta_{g_{i,j}}$. The orbit graph of $\Omega \cdot S$ is defined analogously. A *Schreier tree* for $\alpha \cdot S$ is just a spanning tree for the orbit graph with root at β_1. More precisely, a Schreier tree for $\alpha \cdot S$ is simply a 2-dimensional array

$$\begin{array}{ccc} v_2 & \ldots & v_n \\ w_2 & \ldots & w_n \end{array}$$

such that $\beta_{v_j} \cdot x_{w_j} = \beta_j$ and $v_j < j$ for all $j > 1$.

The orbit graph of $\Omega \cdot S$ may not be connected and so it has a forest of (not necessarily disjoint) Schreier trees rooted at some elements of Ω. To simplify things, we may suppose without loss of generality that there is an $\alpha \in \Omega$ such that $\alpha \cdot S = \Omega \cdot S$. This can be achieved by adding an artificial α to Ω (and perhaps some further points) and defining the action of S on these values so that $\alpha \cdot S$ contains all of the roots of the Schreier forest for $\Omega \cdot S$.

Note that unlike the orbit graph of a component of a group acting on a set, the orbit graph of a component of a semigroup acting on a set is, in general, not strongly connected. This makes several of the steps required below more complicated than in the group case.

Algorithm 1 Compute a component of an action

Input: $S := \langle X \rangle$ where $X := \{x_1, \dots, x_m\}$, S acts on a set Ω on the right, and $\alpha \in \Omega$
Output: $\alpha \cdot S$, a Schreier tree for $\alpha \cdot S$, and the orbit graph of $\alpha \cdot S$

$\alpha \cdot S := \{\beta_1 := \alpha\}$, $n := 1$ [*initialise* $\alpha \cdot S$]
for $\beta_i \in \alpha \cdot S$, $j \in \{1, \dots, m\}$ **do**   [*loop over: existing values in* $\alpha \cdot S$, *the generators*]
    if $\beta_i \cdot x_j \notin \alpha \cdot S$ **then**
        $n := n + 1$, $\beta_n := \beta_i \cdot x_j$, $\alpha \cdot S \leftarrow \alpha \cdot S \cup \{\beta_n\}$;   [*add* $\beta_i \cdot x_j$ *to* $\alpha \cdot S$]
        $v_n := i$, $w_n := j$, $g_{i,j} := n$ [*update the Schreier tree and orbit graph*]
    else if $\beta_i \cdot x_j = \beta_r \in \alpha \cdot S$ **then**
        $g_{i,j} := r$;   [*update the orbit graph*]
    end if
end for
return $\alpha \cdot S$, $(v_2, \dots, v_n, w_2, \dots, w_n)$, $\{g_{i,j} : i \in \{1, \dots, n\},\ j \in \{1, \dots, m\}\}$

The Schreier tree for $\alpha \cdot S$ produced by Algorithm 1 can be used to obtain elements $u_i \in S$ such that $\beta_1 \cdot u_i = \beta_i$ for all i using Algorithm 2. The standard method for a group acting on a set, such as the procedure U-Beta in [45, p80], cannot be used here, due to the non-existence of inverses in semigroups.

Algorithm 2 Trace a Schreier tree

Input: a Schreier tree $(v_2, \dots, v_n, w_2, \dots, w_n)$ for $\alpha \cdot S$, and $\beta_i \in \alpha \cdot S$
Output: $u \in S$ such that $\alpha \cdot u = \beta_i$

$u := 1_S$, $j := i$
while $j > 1$ **do**
    $u := x_{w_j} u$ and $j := v_j$
end while
return u

The right action of S on Ω induces an action of S on $\mathcal{P}(\Omega)$. In Algorithm 3, Algorithms 1 and 2 are used to obtain the Schreier generators from Proposition 2.3(c) for the stabiliser S_Σ of a subset Σ of Ω under this induced action.

Algorithm 3 Compute Schreier generators for a stabiliser

Input: the component $\Sigma \cdot S$ of $\Sigma \subseteq \Omega$ under the action of S on $\mathcal{P}(\Omega)$, a Schreier tree and orbit graph for $\Sigma \cdot S$
Output: Schreier generators Y for the stabiliser S_Σ

$Y := \{\mathrm{id}_\Sigma\}$
find the s.c.c. of Σ in $\Sigma \cdot S := \{\Sigma_1 := \Sigma, \dots, \Sigma_r\}$
for $i \in \{1, \dots, r\}$, $j \in \{1, \dots, m\}$ **do**   [*loop over:* $\Sigma \cdot S$, *the generators of* S]
    set $k := g_{i,j}$ [$g_{i,j}$ *is from the orbit graph of* $\Sigma \cdot S$]
    if $\Sigma_k \sim \Sigma_1$ **then**   [Σ_k *is in the s.c.c. of* Σ_1]
        find $u_i, u_k \in S$ such that $\Sigma_1 \cdot u_i = \Sigma_i$ and $\Sigma_1 \cdot u_k = \Sigma_k$ [*Algorithm 2*]
        find $\overline{u_k} \in U$ such that $\Sigma_k \cdot \overline{u_k} = \Sigma_1$ and $(u_k \overline{u_k})|_{\Sigma_1} = \mathrm{id}_{\Sigma_1}$ [*Proposition 2.3(a)*]
        $Y \leftarrow Y \cup \{(u_i x_j \overline{u_k})|_{\Sigma_1}\}$
    end if
end for
return Y

In Algorithm 4 we give a more specialised version of Algorithm 3, which we require to find $(\mathrm{Stab}_S(L_x^U))\mu_x = S_x$ where μ_x is the function defined at the start of this section.

Algorithm 4 Compute Schreier generators for S_x

Input: the component $(x)\lambda \cdot S$ of $(x)\lambda$, a Schreier tree and orbit graph for $(x)\lambda \cdot S$
Output: Schreier generators Y for the stabiliser S_x

```
set Y := {1_{S_x}}, x_1 := x
find the s.c.c. {(x_1)λ, ..., (x_r)λ} of (x)λ in (x)λ · S
for i ∈ {1, ..., r}, j ∈ {1, ..., m} do        [loop over: the s.c.c. of (x)λ, the generators of S]
    set k := g_{i,j}                            [g_{i,j} is from the orbit graph of (x)λ · S]
    if (x_k)λ ~ (x_1)λ then                     [(x_k)λ is in the s.c.c. of (x_1)λ]
        find u_i, u_k ∈ S such that (x_1)λ · u_i = (x_i)λ and (x_1)λ · u_k = (x_k)λ    [Algorithm 2]
        find ū_k ∈ U such that x u_k ū_k = x    [Assumption (III)]
        Y ← Y ∪ {(x' u_i x_j ū_k)μ_x}           [Assumption (IV)]
    end if
end for
return Y
```

Algorithms 3 and 4 can also be used to find generators for the stabiliser of any value in the component $\Sigma \cdot S$ or $(x)\lambda \cdot S$, if we have a Schreier tree for the s.c.c. rooted at that value. Algorithm 1 returns a Schreier tree for the entire component (possibly including several strongly connected components) rooted at the first point in the component. It is possible to find a Schreier tree for any s.c.c. rooted at any value in the component by finding a spanning tree for the subgraph of the orbit graph the s.c.c. induces. Such a spanning tree can be found in linear time using a depth first search algorithm, for example.

5.4 Individual Green's classes

Data structures

By Corollary 3.15, we can represent the $\mathscr{R}$-class R_x^S of any element $x \in S$ as a quadruple consisting of:

- the representative x;
- the s.c.c. $\{(x)\lambda = \alpha_1, \ldots, \alpha_n\}$ of $(x)\lambda$ under the action of S (this can be found using Algorithms 1 and any algorithm to find the strongly connected components of a digraph);
- a Schreier tree for $\{\alpha_1, \ldots, \alpha_n\}$;
- the stabiliser group S_x found using Algorithm 4.

The $\mathscr{L}^S$-class of x in S can be represented using the analogous quadruple using the s.c.c. of $(x)\rho$, and the group ${}_xS$. The Green's $\mathscr{H}$- and $\mathscr{D}$-classes of x in S are represented using the quadruple for R_x^S and the quadruple for L_x^S.

Size of a Green's class

Having the above data structures, it follows from Corollaries 3.15(b) and 3.16(b) that $|R_x^S| = n \cdot |S_x|$ where n is the length of the s.c.c. of $(x)\lambda$ in $(x)\lambda \cdot S$. Similarly, $|L_x^S|$ is the length of the s.c.c. of $(x)\rho$ multiplied by $|{}_xS|$. Suppose that $x' \in U$ is such that $xx'x = x$. Then, by Proposition 3.13(b), $|H_x^S| = |S_x \cap ({}_xS)\Psi|$, where $\Psi : {}_xS \longrightarrow U_x$ is defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$ for all $s \in {}_xS$. The group ${}_xS$ can be found using the analogue of Algorithm 4, and we can compute the values $(x'sx)\mu_x$ for all $s \in {}_xS$ by Assumption (V). The size of the $\mathscr{D}$-class D_x^S is just

$$|D_x^S| = \frac{|L_x^S| \cdot |R_x^S|}{|H_x^S|},$$

and so $|D_x^S|$ can be found using the values of $|L_x^S|$, $|R_x^S|$, and $|H_x^S|$.

Elements of a Green's class

Corollary 3.20(a) states that

$$R_x^S = \{xsu : s \in \mathrm{Stab}_S(L_x^U),\ u \in S,\ (x)\lambda \cdot u \sim (x)\lambda\}.$$

If $s, t \in \mathrm{Stab}_S(L_x^U)$ are such that $(s)\mu_x = (t)\mu_x$, then, by Lemma 3.8, $xs = xt$. It follows that if M is any subset of $\mathrm{Stab}_S(L_x^U)$ such that $(M)\mu_x = S_x$ and $(s)\mu_x \neq (t)\mu_x$ for all $s, t \in M$ such that $s \neq t$, then

$$R_x^S = \{xsu : s \in M,\ u \in S,\ (x)\lambda \cdot u \sim (x)\lambda\}.$$

If $Y = \{(y_1)\mu_x, \ldots, (y_k)\mu_x\}$ is a set of generators for S_x (from Algorithm 4), then every element of S_x is of the form $(s)\mu_x$ where $s \in \langle y_1, \ldots, y_k\rangle$. Thus the set M can be found by computing the elements of S_x, and expressing each element as a product of the generators Y. An algorithm for finding the elements of R_x^S is given in Algorithm 5.

Algorithm 5 Elements of an $\mathscr{R}$-class

Input: $x \in S$
Output: the elements Y of the $\mathscr{R}$-class R_x^S

$Y := \varnothing$
find the s.c.c. $\{(x)\lambda = \alpha_1, \ldots, \alpha_n\}$ of $(x)\lambda$, and a Schreier tree for this s.c.c. [*Algorithm 1*]
find the group S_x [*Algorithm 4*]
for $i \in \{1, \ldots, n\}$, $(s)\mu_x \in S_x$ **do** [*Loop over: the s.c.c., elements of the group*]
    find $u_i \in S$ such that $\alpha_1 \cdot u_i = \alpha_i$ [*Algorithm 2*]
    $Y \leftarrow Y \cup \{xsu_i\}$
end for
return Y

The elements of an $\mathscr{L}$-class can be found using an analogous algorithm. By the proof of Proposition 3.13(b), $\phi_1 : S_x \cap ({}_xS)\Psi \longrightarrow H_x^S$ defined by $((s)\mu_x)\phi_1 = xs$ is a bijection. It follows that if we can compute the intersection of groups $S_x \cap ({}_xS)\Psi$, then we can obtain the elements of H_x^S. As mentioned above, $({}_xS)\Psi$ can be determined using Algorithm 4 and by Assumption (V).

The elements of a $\mathscr{D}$-class are slightly more complicated to compute. In Algorithm 6 we show how to find the $\mathscr{R}$-classes in a given $\mathscr{D}$-class of S and this combined with Algorithm 5 gives a method for finding the elements of a $\mathscr{D}$-class.

Classes within classes

By Proposition 3.23, if $x \in S$, $x' \in U$ is such that $xx'x = x$, and:

- $\mathcal{C}$ is a subset of $\mathrm{Stab}_U(L_x^U)$ such that $\{(c)\mu_x : c \in \mathcal{C}\}$ is a left transversal of $S_x \cap ({}_xS)\Psi$ in $({}_xS)\Psi$ where $\Psi : {}_xS \longrightarrow U_x$, defined by $((s)\nu_x)\Psi = (x'sx)\mu_x$, is the embedding from Proposition 3.13(a);
- $\{u_i \cdot (x)\rho : 1 \leq i \leq m\}$ is the s.c.c. of $(x)\rho$ under the left action of S, where $u_i \in S$ for all i,

then $\{u_ixc : c \in \mathcal{C},\ 1 \leq i \leq m\}$ is a set of $\mathscr{H}^S$-class representatives for L_x^S, and hence a set of $\mathscr{R}^S$-class representatives for D_x^S. Using this result in Algorithm 6 we show how to find the $\mathscr{R}^S$-classes of a $\mathscr{D}^S$-class. Since $(u_ixc)\lambda = (x)\lambda$, it follows that $S_{u_ixc} = S_x$ for all i and all $c \in \mathcal{C}$. Therefore, the data structures for R_x^S and $R_{u_ixc}^S$ are identical except for the representatives.

From Proposition 3.24, algorithms analogous to Algorithm 6, can be used to find the $\mathscr{L}^S$-classes in a $\mathscr{D}^S$-class, the $\mathscr{H}^S$-classes in an $\mathscr{R}^S$-class, or the $\mathscr{H}^S$-classes in an $\mathscr{L}^S$-class.

Testing membership

Using Corollary 3.18 and 3.22, in Algorithms 7 and 8 we show how the data structures described at the start of the section can be used to test membership in an $\mathscr{R}$- or $\mathscr{D}$-class.

Using Proposition 3.19, an algorithm analogous to Algorithm 7 (for $\mathscr{R}$-classes) can be used to test membership in an $\mathscr{L}$-class. Testing membership in an $\mathscr{H}$-class can then be accomplished by testing membership in the corresponding $\mathscr{L}$- and $\mathscr{R}$-classes.

Algorithm 6 $\mathscr{R}$-classes in a $\mathscr{D}$-class

Input: $x \in S$
Output: $\mathscr{R}$-class representatives $\mathfrak{R}$ of the $\mathscr{D}$-class D_x^S

$\mathfrak{R} := \varnothing$
find the s.c.c.s of $(x)\lambda$ and $(x)\rho$ and their Schreier trees [*Algorithm 1*]
find $S_x \cap ({}_xS)\Psi$ [*Algorithm 4 and Assumption* (V)]
find $\mathcal{C} \subseteq \mathrm{Stab}_S(L_x^U)$ such that $\{(c)\mu_x : c \in \mathcal{C}\}$ is a left transversal of $S_x \cap ({}_xS)\Psi$ in $({}_xS)\Psi$ [*Proposition 3.23(a)*]
for $(y)\rho$ in the s.c.c. of $(x)\rho$ **do**
    find $u_i \in S$ such that $u_i \cdot (x)\rho = (y)\rho$ [*Algorithm 2*]
    for $c \in \mathcal{C}$ **do**
        $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{u_i x c\}$
    end for
end for
return $\mathfrak{R}$

Algorithm 7 Test membership in an $\mathscr{R}$-class

Input: $y \in U$ and the data structure of an $\mathscr{R}$-class R_x^S
Output: true or false

if $(x)\rho = (y)\rho$ and $(y)\lambda \sim (x)\lambda$ **then**
    find $u \in S$ such that $(x)\lambda \cdot u = (y)\lambda$ [*Algorithm 2*]
    find $\overline{u} \in U$ such that $(y)\lambda \cdot \overline{u} = (x)\lambda$ and $xu\overline{u} = x$ [*Assumption* (III)]
    return $(x'y\overline{u})\mu_x \in S_x$ [*Assumption* (IV)]
else
    return false
end if

Algorithm 8 Test membership in a $\mathscr{D}$-class

Input: $y \in U$ and the data structure for the $\mathscr{D}$-class of $x \in S$
Output: true or false

if $(x)\lambda \sim (y)\lambda$ and $(x)\rho \sim (y)\rho$ **then**
    find $u_1, u_2 \in S$ such that $(x)\lambda \cdot u_1 = (y)\lambda$ and $u_2 \cdot (x)\rho = (y)\rho$ [*Algorithm 2*]
    find $\overline{u_1} \in U$ such that $(y)\lambda \cdot \overline{u_1} = (x)\lambda$ and $xu_1\overline{u_1} = x$ [*Assumption* (III)]
    find $\overline{u_2} \in U$ such that $\overline{u_2} \cdot (y)\rho = (x)\rho$, and $\overline{u_2}u_2x = x$ [*the analogue of Assumption* (III)]
    find $S_x \cap ({}_xS)\Psi$ [*Algorithm 4 and Assumption* (V)]
    find $\mathcal{C} \subseteq S$ such that $\{(c)\mu_x : c \in \mathcal{C}\}$ is a left transversal of $S_x \cap ({}_xS)\Psi$ in S_x [*Proposition 3.23(a)*]
    for $c \in \mathcal{C}$ **do**
        if $(x'\overline{u_2}y\overline{u_1}c)\mu_x$ in $({}_xS)\Psi$ **then**
            return true
        end if
    end for
end if
return false

Regularity and idempotents

An $\mathscr{R}$-class R of S is regular if and only if there is $x \in R$ such that H_x^U is a group. Since $y \in H_x^U$ if and only if $(x)\lambda = (y)\lambda$ and $(x)\rho = (y)\rho$, it is possible to verify that an $\mathscr{R}$-class is regular only by considering the value $(x)\lambda$ and the s.c.c. of $(x)\rho$. A similar approach can be used to to compute the idempotents in an $\mathscr{R}$-class.

How we test if the $\mathscr{H}$-class in U corresponding to $(x)\lambda$ and $(y)\rho$ is a group, depends on the context. For example:

Transformation semigroups: in the full transformation monoid, $(x)\lambda$ and $(y)\rho$ are an image set and kernel of a transformation, respectively. In this case, the $\mathscr{H}$-class corresponding to $(x)\lambda$ and $(y)\rho$ is a group if and only if $(x)\lambda$ contains precisely one element in every kernel class of $(x)\rho$. If $(x)\lambda$ and $(y)\rho$ satisfy this property, then it is relatively straightforward to compute an idempotent with kernel $(y)\rho$ and image $(x)\lambda$;

Partial permutations: in the symmetric inverse monoid, $(x)\lambda$ and $(y)\rho$ are the image set and domain of a partial permutation, respectively. In this case, the $\mathscr{H}$-class corresponding to $(x)\lambda$ and $(y)\rho$ is a group if and only if $(x)\lambda = (y)\rho$. Given such $(x)\lambda = (y)\rho$, the partial identity function with domain $(x)\lambda$ is the idempotent in the $\mathscr{H}$-class;

Matrix semigroups: In this case, there is no simple criteria for the $\mathscr{H}$-class of $(x)\lambda$ and $(y)\rho$ to be a group.

Rees matrix semigroups: in a Rees 0-matrix semigroup $\mathcal{M}^0[G; I, J; P]$, $(x)\lambda = j \in J$ and $(y)\rho = k \in I$. In this case, the $\mathscr{H}$-class corresponding to $(x)\lambda$ and $(y)\rho$ is a group if and only if $p_{j,k} \neq 0$. The idempotent in the $\mathscr{H}$-class is then $(k, p_{j,k}^{-1}, j)$;

Partitions: in the partition monoid, $(x)\lambda = x^*x$ and $(y)\rho = yy^*$. In this case, the $\mathscr{H}$-class $L_{x^*x} \cap R_{yy^*}$ contains an idempotent if and only if the number of transverse blocks of yy^*x^*x equals the number of transverse blocks of x^*x. If the previous condition is satisfied, then the idempotent contained in $L_{x^*x} \cap R_{yy^*}$ is yy^*x^*x.

Algorithm 9 Regularity of an $\mathscr{R}$-class

Input: a representative $x \in S$ of an $\mathscr{R}$-class of S
Output: true or false

```
find the s.c.c. of (x)ρ in S · (x)ρ                    [analogue of Algorithm 1]
for (y)ρ in the s.c.c. of (x)ρ do
    if the ℋ-class in U corresponding to (x)λ and (y)ρ is a group then
        return true
    end if
end for
return false
```

Algorithms analogous to Algorithms 9 and 10 can be used to test regularity and find the idempotents in an $\mathscr{L}$-class. A $\mathscr{D}$-class D in S is regular if and only if any (equivalently, every) $\mathscr{R}$-class in D is regular. Hence Algorithms 6 and 9 and can be used to verify if a $\mathscr{D}$-class is regular or not. It is possible to calculate the idempotents in a $\mathscr{D}$-class D by creating the $\mathscr{R}$-class representatives using Algorithm 6 and finding the idempotents in every $\mathscr{R}$-class of D using Algorithm 10.

5.5 The global structure of a semigroup

In this section, we provide algorithms for determining the structure of an entire semigroup, rather than just its individual Green's classes as in the previous sections. Unlike the previous two subsections, the algorithms described in this section differ significantly from the analogous procedures described in [24].

The main algorithm

Algorithm 11 is the main algorithm for computing the size, the Green's structure, testing membership, and so on in S. This algorithm could be replaced by an analogous algorithm which enumerates $\mathscr{L}$-classes,

Algorithm 10 Idempotents in an $\mathscr{R}$-class

Input: $x \in S$
Output: the idempotents E of R_x^S

```
E := ∅
find the s.c.c. of (x)ρ in S · (x)ρ                    [analogue of Algorithm 1]
for (y)ρ in the s.c.c. of (x)ρ do
    if the ℋ-class H in U corresponding to (x)λ and (y)ρ is a group then
        find the identity e of H
        E ← E ∪ {e}
    end if
end for
return E
```

rather than $\mathscr{R}$-classes. In the case of transformation semigroups, in many test cases, the algorithm for $\mathscr{R}$-classes has better performance than the analogous algorithm for $\mathscr{L}$-classes. This is one reason for presenting this algorithm and not the other.

Since Green's $\mathscr{R}$-relation is a left congruence, it follows that representatives of the $\mathscr{R}$-classes of S can be obtained from the identity by left multiplying by the generators. The principle purpose of Algorithm 11 is to determine the action of S on its $\mathscr{R}$-class representatives by left multiplication. In particular, we enumerate $\mathscr{R}$-class representatives of S, which we will denote by $\mathfrak{R}$. Since we are calculating an action of S, we may also discuss the Schreier tree and orbit graph of this action, as we did in Algorithm 1. At the same time as finding the action of S on its $\mathscr{R}$-class representatives, we also calculate $(S)\rho$. Since $(x)\rho = (y)\rho$ if and only if $x\mathscr{R}^U y$, it follows that

$$(S)\rho = (\mathfrak{R})\rho = \{(x)\rho : x \in \mathfrak{R}\}.$$

In Algorithm 11, we find an addition parameter, which is denoted by K_i. This parameter is used later in Algorithm 13, which allows us to factorise elements of a semigroup over its generators.

If the subsemigroup S we are trying to compute is $\mathscr{R}$-trivial, then Algorithm 11 simply exhaustively enumerates the elements of S. In such a case, unfortunately, Algorithm 11 has poorer performance than a well-implemented exhaustive algorithm, since it contains some superfluous steps. For example, the calculations of $(S)\lambda$ and the groups S_x are unnecessary steps if S is $\mathscr{R}$-trivial. On a more positive note, in some cases, it is possible to detect if a semigroup is $\mathscr{R}$-trivial with relatively little effort. For example, if S is a transformation semigroup of degree n, then it is shown in [40] that S is $\mathscr{R}$-trivial if and only if its action on the points in $\{1, \ldots, n\}$ is acyclic. In other words, it is possible to check whether a transformation semigroup is $\mathscr{R}$-trivial in polynomial time $O(n^3)$. At least in this case, it would then be possible to use the $\mathscr{L}$-class version of Algorithm 11 instead of the $\mathscr{R}$-class version, or indeed, an exhaustive algorithm such as that in [34].

From Algorithm 11, it is routine to calculate the size of S as:

$$|S| = \sum_{i=1}^{r} |S_{z_i}| \cdot |\text{s.c.c. of } (z_i)\lambda| \cdot |\{y \in \mathfrak{R} : (y)\lambda = (z_i)\lambda\}|. \tag{5.1}$$

The elements of S are just the union of the sets of elements of the $\mathscr{R}$-classes of S, which can be found using Algorithm 5.

The idempotents of S can be found by determining the idempotents in the $\mathscr{R}$-classes of S using Algorithm 10. Similarly, it is possible to test if S is regular, by checking that every $\mathscr{R}$-class or $\mathscr{D}$-class is regular using Algorithm 9. Note that the existence of an idempotent in an $\mathscr{R}^S$-class depends only on the values of λ and ρ of the elements of that class. Hence if there are at least two distinct $\mathscr{R}$-class representatives x and y in S such that $(x)\lambda = (y)\lambda$ and $(x)\rho = (y)\rho$, then S is not regular, since the disjoint $\mathscr{R}$-classes R_x^S and R_y^S cannot contain the same idempotents.

The $\mathscr{D}$-classes of S are in 1-1 correspondence with the strongly connected components of the orbit graph of $\mathfrak{R}$, which is obtained in Algorithm 11. Since we also find $(S)\rho$ in Algorithm 11, it is possible to find the $\mathscr{D}$-classes of S, and their data structures, by finding the strongly connected components of the orbit graph of $\mathfrak{R}$. The $\mathscr{L}$- and $\mathscr{H}$-classes of S can then be found from the $\mathscr{D}$-classes using the analogues of Algorithm 6 (for finding the $\mathscr{R}$-classes in a $\mathscr{D}$-class).

Algorithm 11 Enumerate the $\mathscr{R}$-classes of a semigroup

Input: $S := \langle X \rangle$ where $X := \{x_1, \ldots, x_m\}$

Output: data structures for the $\mathscr{R}$-classes with representatives $\mathfrak{R}$ in S, the set $(S)\rho$, and Schreier trees and orbit graphs for $\mathfrak{R}$ and $(S)\rho$

1: set $\mathfrak{R} := \{y_1 := 1_S\}$, $M := 1$ [*initialise the list of $\mathscr{R}$-class reps*]
2: set $(S)\rho := \{\beta_1 := (1_S)\rho\}$, $N := 1$ [*initialise $(S)\rho$*]
3: find $(S)\lambda = (1_S)\lambda \cdot S$ [*Algorithm 1*]
4: find representatives $(z_1)\lambda, \ldots, (z_r)\lambda$ of the s.c.c.s of $(S)\lambda$ [*standard graph theory algorithm*]
5: find the groups S_{z_i} for $i \in \{1, \ldots, r\}$ [*Algorithm 4*]
6: **for** $y_i \in \mathfrak{R}$, $j \in \{1, \ldots, m\}$ **do** [*loop over: existing $\mathscr{R}$-representatives, generators of S*]
7:     find $n \in \{1, \ldots, r\}$ such that $(z_n)\lambda \sim (x_j y_i)\lambda$
8:     find $u \in S$ such that $(z_n)\lambda \cdot u = (x_j y_i)\lambda$ [*Algorithm 2*]
9:     find $\overline{u} \in U$ such that $(x_j y_i)\lambda \cdot \overline{u} = (z_n)\lambda$ and $z_n u \overline{u} = z_n$ [*Assumption (III)*]
10:     **if** $(x_j y_i)\rho \notin (S)\rho$ **then** [*$x_j y_i \overline{u}$ is a new $\mathscr{R}$-representative*]
11:         $N := N + 1$, $\beta_N := (x_j y_i \overline{u})\rho = x_j \cdot (y_i)\rho$, $(S)\rho \leftarrow (S)\rho \cup \{\beta_N\}$ [*add $(x_j y_i)\rho$ to $(S)\rho$*]
12:         $v_N := i$, $w_N := j$, $g_{i,j} := N$ [*update the Schreier tree and orbit graph of $(S)\rho$*]
13:     **else if** $(x_j y_i)\rho = \beta_k \in (S)\rho$ **then**
14:         $g_{l,j} := k$ where l is such that $(y_i)\rho = \beta_l$; [*update the orbit graph of $(S)\rho$*]
15:         **for** $y_l \in \mathfrak{R}$ with $(y_l)\rho = (x_j y_i)\rho$ and $(y_l)\lambda = (z_n)\lambda$ **do**
16:             **if** $(y_l' x_j y_i \overline{u})\mu_{z_n} \in S_{z_n} = S_{y_l}$ **then** [$x_j y_i \mathscr{R}^S y_l$]
17:                 $G_{i,j} := l$ [*update the orbit graph of $\mathfrak{R}$*]
18:                 go to line 6
19:             **end if**
20:         **end for**
21:     **end if**
22:     $M := M + 1$, $y_M := x_j y_i \overline{u}$, $\mathfrak{R} \leftarrow \mathfrak{R} \cup \{y_M\}$ [*add $x_j y_i \overline{u}$ to $\mathfrak{R}$*]
23:     $V_M := i$, $W_M := j$, $K_M :=$ the index of $(x_j y_i)\lambda$ in $(S)\lambda$ [*update the Schreier tree of $\mathfrak{R}$*]
24:     $G_{i,j} := M$ [*update the orbit graph of $\mathfrak{R}$*]
25: **end for**
26: **return** the following:

- the $\mathscr{R}$-representatives: $\mathfrak{R}$
- the Schreier tree for $\mathfrak{R}$: $(V_2, \ldots, V_M, W_2, \ldots, W_M)$
- the orbit graph of $\mathfrak{R}$: $\{G_{i,j} : i = 1, \ldots, M,\ j = 1, \ldots, m\}$
- the set $(S)\rho$
- the Schreier tree for $(S)\rho$: $(v_2, \ldots, v_N, w_2, \ldots, w_N)$
- the orbit graph of $(S)\rho$: $\{g_{i,j} : i = 1, \ldots, N,\ j = 1, \ldots, m\}$
- the parameters: $(K_1, \ldots, K_M)$

In Algorithms 12, 13, 14, and 15, we will assume that Algorithm 11 has been performed and that the result has been stored somehow. So, for example, if we wanted to check membership of several elements in the semigroup U in its subsemigroup S, we perform Algorithm 11 only once.

Testing membership in a semigroup

In Algorithm 12, we give a procedure for testing if an element of U belongs to S. This can be easily modified to return the $\mathscr{R}$-class representative in S of an arbitrary element of U, if it exists (simply return the element x in line 12). We require such an algorithm when it comes to factorising elements of S over the generators in Algorithm 13.

Algorithm 12 Test membership in a semigroup

Input: $S := \langle X \rangle$ where $X := \{x_1, \ldots, x_m\}$ and $y \in U$
Output: true or false

1: let $(S)\lambda = (1)\lambda \cdot S$ and $(S)\rho = S \cdot (1)\rho$ [*Algorithm 1*]
2: **if** $(y)\lambda \notin (S)\lambda$ or $(y)\rho \notin (S)\rho$ **then** $[y \notin S]$
3:     **return** false
4: **end if**
5: let $(z_1)\lambda, \ldots, (z_r)\lambda$ be representatives of the s.c.c.s of $(S)\lambda$
6: let $\mathfrak{R}$ denote the $\mathscr{R}$-class representatives of S [*Algorithm 11*]
7: find $n \in \{1, \ldots, r\}$ such that $(z_n)\lambda \sim (y)\lambda$
8: find $u \in S$ such that $(z_n)\lambda \cdot u = (y)\lambda$ [*Algorithm 2*]
9: find $\overline{u} \in U$ such that $(y)\lambda \cdot \overline{u} = (z_n)\lambda$ and $yu\overline{u} = y$ [*Assumption* (III)]
10: **for** $x \in \mathfrak{R}$ such that $(x)\lambda = (z_n)\lambda$ and $(x)\rho = (y)\rho$ **do**
11:     **if** $(x'y\overline{u})\mu_x \in S_{z_n} = S_x$ **then** $[x\mathscr{R}^S y]$
12:         **return** true
13:     **end if**
14: **end for**
15: **return** false

Factorising elements over the generators

In this part of the paper, we describe how to factorise an element of S as a product of the generators of S using the output of Algorithm 11.

Corollary 3.20(a) says that

$$R_x^S = \{xsu : s \in \mathrm{Stab}_S(L_x^U),\ u \in S,\ (xu)\lambda \sim \lambda(x)\}.$$

Suppose that $y \in S$ is arbitrary and that $\mathfrak{R}$ denotes a set of $\mathscr{R}$-class representatives of S. If we can write $y = xsu$, where $x \in \mathfrak{R}$ such that $x\mathscr{R}^S y$, $s \in \mathrm{Stab}_S(L_x^U)$, and $u \in S$ such that $(xu)\lambda = (y)\lambda$, then it suffices to factorise each of x, s, and u individually.

The element $x \in \mathfrak{R}$ can be found using the alternate version of Algorithm 12 mentioned above. Algorithm 2, applied to $(S)\lambda$, can be used to find u such that $(xu)\lambda = (y)\lambda$.

Suppose that $\overline{u} \in U$ is such that $(y)\lambda \cdot \overline{u} = (x)\lambda$ and $xu\overline{u} = x$. Since $(xs)\lambda = (x)\lambda$, it follows from Lemma 3.8 that $xsu\overline{u} = xs$. If $x' \in U$ is any element such that $xx'x = x$, then from Proposition 3.11(a), x is the identity of the group $L_x^U \cap R_x^S$ under multiplication $*$ defined by $a * b = ax'b$. In particular, $xx'y\overline{u} = y\overline{u}$ since $y\overline{u} \in L_x^U \cap R_x^S$. This implies that

$$xx'y\overline{u} = y\overline{u} = xsu\overline{u} = xs$$

and so, by Lemma 3.8, $(x'y\overline{u})\mu_x = (s)\mu_x$. Since $(y\overline{u})\lambda = (x)\lambda$ and $(y\overline{u})\rho = (x)\rho$, by Assumption (IV), we can compute $(x'y\overline{u})\mu_x$. Thus for any $y \in S$ we can determine $x, s, u \in S$ (as above) such that $y = xsu$.

We still require a factorisation of $x, s, u \in S$ over the generators of S. Tracing the Schreier tree of $\mathfrak{R}$ returned by Algorithm 11 and using the parameter K_i, we can factorise $x \in \mathfrak{R}$; more details are in Algorithm 13. We may factorise u over the generators of S using Algorithm 2 applied to the s.c.c. of $(x)\lambda$ in $(S)\lambda$.

Any algorithm for factorising elements of a group can be used to factorise $(s)\mu_x$ as a product of the generators of S_x given by Algorithm 4. For example, in a group with a faithful action on some set, such a factorisation can be obtained from a stabiliser chain, produced using the Schreier-Sims algorithm. The generators of S_x are of the form $(u_i x_k \overline{u_j})\mu_x$, where u_i is obtained by tracing the Schreier tree of $(S)\lambda$, x_k is one of the generators of S, and $\overline{u_j}$ is obtained using Assumption (III). Therefore to factorise s over the generators of S, it suffices to factorise the $\overline{u_j}$ over the generators of S.

Suppose that $\{\alpha_1, \ldots, \alpha_K\}$ is a s.c.c. of $(S)\lambda$ for some $K \in \mathbb{N}$. Then from the orbit graph of $(S)\lambda$ we can find

$$a_2, \ldots, a_K, b_2, \ldots, b_K$$

such that

$$\alpha_j \cdot x_{a_j} = \alpha_{b_j}$$

and $b_j < j$ for all $j > 1$. In other words, $(a_2, \ldots, a_K, b_2, \ldots, b_K)$ describes a spanning tree, rooted at α_1, for the component of the orbit graph of $(S)\lambda$, whose edges have the opposite orientation to those in the usual Schreier tree. We refer to $(a_2, \ldots, a_K, b_2, \ldots, b_K)$ as a *reverse Schreier tree.*

It follows that for any $i \in \{1, \ldots, K\}$ we can use an analogue of Algorithm 2 to obtain $v \in S$ such that $\alpha_j \cdot v = \alpha_1$. However, if $u \in S$ is obtained using Algorithm 2 such that $\alpha_1 \cdot u = \alpha_j$ and $z \in S$ is such that $(z)\lambda = \alpha_1$, then it is possible that $(uv)\mu_z \neq (1_U)\mu_z$. So, if $w \in \mathrm{Stab}_S(L_z^U)$ is such that $(w)\mu_z = ((uv)\mu_z)^{-1}$, then $\alpha_j \cdot vw = \alpha_1$ $(uvw)\mu_z = (1_U)\mu_z$. We have factorisations of v, since it was obtained by tracing the reverse Schreier tree, and w, since it can be given as a power of uv (u and v are factorised), over the generators of S. Hence vw is factorized over the generators of S, and it has the properties required of $\overline{u_j}$ from the previous paragraph.

We note that all the information required to factorise any element of S is returned by Algorithm 11 except the factorisation of $(s)\mu_x$ in S_x. So, Algorithm 13 is just concerned with putting this information together. There is no guarantee that the word produced by Algorithm 13 is of minimal length.

Algorithm 13 Factorise an element over the generators

Input: $S := \langle X \rangle$ where $X := \{x_1, \ldots, x_m\}$ and $s \in S$
Output: a word in the generators X equal to s

let $\theta : X^+ \longrightarrow S$ be the unique homomorphism extending the inclusion of X in S
suppose that $(S)\lambda := \{(z_1)\lambda, \ldots, (z_K)\lambda\}$ for some $z_1, \ldots, z_K \in S$ [*Algorithm 1*]
let $\mathfrak{R} := \{y_1, \ldots, y_r\}$ be the $\mathscr{R}$-representatives of S [*Algorithm 11*]
let $(V_2, \ldots, V_r, W_2, \ldots, W_r)$ be the Schreier tree for $\mathfrak{R}$ [*Algorithm 11*]
let $(K_1, \ldots, K_r)$ denote the additional parameter return from Algorithm 11
find $y_i \in \mathfrak{R}$ such that $y_i \mathscr{R}^S s$ [*the modified version of Algorithm 7*]
find a word $\omega_1 \in X^+$ such that $(\omega_1)\theta = u \in S$ where $(y_i)\lambda \cdot u = (s)\lambda$ [*factorise u using Algorithm 2*]
find $\overline{u} \in U$ such that $(s)\lambda \cdot \overline{u} = (y_i)\lambda$, and $y_i u \overline{u} = y_i$ [*Assumption* (III)]
compute $(y_i' s \overline{u})\mu_{y_i} \in S_{y_i}$ [*Assumption* (IV)]
find a word $\omega_2 \in X^+$ such that $(\omega_2)\theta = y_i' s \overline{u}$ [*factorise s*]
[*factorise $(y_i' s \overline{u})\mu_{y_i}$ over the generators of S_{y_i} and then factorise these generators over X*]
$\omega_3 := \varepsilon$ (the empty word), $j = i$ [*trace the Schreier tree of $\mathfrak{R}$, factorise y_i*]
while $j > 1$ **do**
  find $\beta \in X^+$ such that $(z_{K_j})\lambda \cdot (\beta)\theta = (y_j)\lambda$
  set $\omega_3 := x_{W_j} \omega_3 \beta$ and $j := V_j$
end while [$(\omega_3)\theta = y_i$]
return $\omega_3 \omega_2 \omega_1$ [$(\omega_3\omega_2\omega_1)\theta = y_i y_i' s \overline{u} u = s$]

The partial order of the $\mathscr{D}$-classes

Recall that there is a partial order $\leq_{\mathscr{D}}$ on the $\mathscr{D}$-classes of a finite semigroup S, which is induced by containment of principal two-sided ideals. More precisely, if A and B are $\mathscr{D}$-classes of S, then we write $A \leq_{\mathscr{D}} B$ if $S^1 a S^1 \subseteq S^1 b S^1$ for any (and every) $a \in A$ and $b \in B$.

The penultimate algorithm in this paper allows us to calculate the partial order of the $\mathscr{D}$-classes of S. This algorithm is based on [24, Algorithm Z] and the following proposition which appears as Proposition 5.1 in [20]. The principal differences between our algorithm and Algorithm Z in [24] are that our algorithm applies to classes of semigroups other than transformation semigroups, and it takes advantage of information already determined in Algorithm 11.

Proposition 5.2 (cf. Proposition 5.1 in [20])**.** *Let S be a finite semigroup generated by a subset X, and let D be a $\mathscr{D}$-class of S. If R and L are representatives of the $\mathscr{R}$- and $\mathscr{L}$-classes in D, then the set $XR \cup LX$ contains representatives for the $\mathscr{D}$-classes immediately below D under $\leq_{\mathscr{D}}$.*

We described above that we find the $\mathscr{D}$-classes (or representatives for the $\mathscr{D}$-classes) of S by finding the s.c.c.s of the orbit graph of the $\mathscr{R}$-class representatives $\mathfrak{R}$ of S. Thus, after performing Algorithm 11 and finding the s.c.c.s of the orbit graph of $\mathfrak{R}$, we know both the $\mathscr{D}$-classes of S and the $\mathscr{R}$-classes contained in the $\mathscr{D}$-classes. In Algorithm 11, we obtain the $\mathscr{R}$-class representatives of S by left multiplying existing representatives. Therefore we have already found the information required to determine the set XR in Proposition 5.2. In particular, we do not have to multiply the $\mathscr{R}$-class representatives of a $\mathscr{D}$-class by each of the generators in Algorithm 14, or determine which $\mathscr{D}$-classes correspond to the elements of XR.

In Algorithm 14 we represent the partial order of the $\mathscr{D}$-classes $D_1, \ldots, D_n$ of S as $P_1, \ldots, P_n$ where P_i contains the indices of the $\mathscr{D}$-classes immediately below D_i (and maybe some more). The elements of P_i are obtained by applying Proposition 5.2 to D_i.

Algorithm 14 The partial order of the $\mathscr{D}$-classes of a semigroup

Input: $S := \langle X \rangle$ where $X := \{x_1, \ldots, x_m\}$
Output: the partial order of the $\mathscr{D}$-classes of S

let $\mathfrak{R}$ denote the $\mathscr{R}$-representatives of S [*Algorithm 11*]
let $\Gamma := \{G_{i,j} : 1 \leq i \leq |\mathfrak{R}|,\ 1 \leq j \leq m\}$ be the orbit graph of $\mathfrak{R}$ [*Algorithm 11*]
find the $\mathscr{D}$-classes $D_1, \ldots, D_n$ of S and $D_i \cap \mathfrak{R}$ for all i [*find the s.c.c.s of* Γ]
let $P_i := \varnothing$ for all i [*initialise the partial order*]
for $i \in \{1, \ldots, n\}$ **do** [*loop over the $\mathscr{D}$-classes*]
  for $y_j \in \mathfrak{R} \cap D_i$, $x_k \in X$ **do** [*loop over: $\mathscr{R}$-classes of the $\mathscr{D}$-class, generators of S*]
    find $l \in \{1, \ldots, n\}$ such that $y_{G_{j,k}} \in D_l$ [*use the orbit graph of* $\mathfrak{R}$]
    $P_i \leftarrow P_i \cup \{l\}$
  end for
  find the $\mathscr{L}$-class representatives $\mathfrak{L}$ in D_i [*the analogue of Algorithm 6, Proposition 3.24*]
  for $z_j \in \mathfrak{L}$, $x_k \in X$ **do** [*loop over: $\mathscr{L}$-classes of the $\mathscr{D}$-class, generators of S*]
    find l such that $z_j x_k \in D_l$ [*use the analogue of Algorithm 7 to find the $\mathscr{R}$-rep. of $z_j x_k$*]
    $P_i \leftarrow P_i \cup \{l\}$
  end for
end for
return $P_1, \ldots, P_n$.

The closure of a semigroup and some elements

The final algorithm (Algorithm 15) we present describes a method for taking the closure of a subsemigroup S of U with a set V of elements of U. The purpose of this algorithm is to reuse whatever information is known about S to make subsequent computations involving $\langle S, V \rangle$ more efficient.

At the beginning of this procedure we form $(\langle S, V \rangle)\lambda$ by adding the new generators V to $(S)\lambda$. This can be achieved in GAP using the `AddGeneratorsToOrbit` function from the Orb package [30], which has the advantage that the existing information in $(S)\lambda$ is not recomputed. The orbit $(S)\lambda$ is extended by a breadth-first enumeration with the new generators without reapplying the old generators to existing values in $(S)\lambda$.

5.6 Optimizations for regular and inverse semigroups

Several of the algorithms presented in this section become more straightforward if it is known *a priori* that the subsemigroup S of U is regular. For example, the $\mathscr{R}$-classes of S are just the $\mathscr{R}$-classes of U intersected with S, and so are in 1-1 correspondence with $(S)\rho$. The algorithms become simpler still under the assumption that U is an inverse semigroup since in this case we may define $(x)\rho = (x^{-1})\lambda$ for all $x \in U$ and so it is unnecessary to calculate $(S)\rho$ and $(S)\lambda$ separately.

The first algorithm where an advantage can be seen is Algorithm 6. Suppose that S is a regular subsemigroup of U, and $x \in S$. Then, by Corollary 3.14, $S_x = ({}_xS)\Psi$ and so it is no longer necessary to compute $({}_xS)\Psi$ or the cosets of $S_x \cap ({}_xS)\Psi$ in $({}_xS)\Psi$ in Algorithm 6. If U is an inverse semigroup with

Algorithm 15 The closure of a semigroup and some elements

Input: $S := \langle X \rangle$ where $X := \{x_1, \dots, x_m\}$ and any existing data structures for S, and $V \subseteq U$

Output: data structures for the $\mathscr{R}$-classes with representatives $\mathfrak{R}$ in $\langle S, V \rangle$, the set $(\langle S, V \rangle)\rho$, and Schreier trees and orbit graphs for $\mathfrak{R}$ and $(\langle S, V \rangle)\rho$

1: set $\widehat{\mathfrak{R}} := \{\widehat{y_1}, \dots, \widehat{y_k}\}$ to be the $\mathscr{R}$-class representatives of S [*Algorithm 11*]
2: set $\mathfrak{R} := \{y_1 := \widehat{y_1} = 1_S\}$, $M := 1$ [*initialise the list of new $\mathscr{R}$-reps*]
3: define $(1)\iota = 1$ [*keep track of indices of old and new $\mathscr{R}$-reps*]
4: set $(\langle S, V \rangle)\rho := (S)\rho$, $N := |(S)\rho|$ [*initialise $(\langle S, V \rangle)\rho$*]
5: set the Schreier tree of $(\langle S, V \rangle)\rho$ to be that of $(S)\rho$ [*initialise the Schreier tree of $(\langle S, V \rangle)\rho$*]
6: set the orbit graph of $(\langle S, V \rangle)\rho$ to be that of $(S)\rho$ [*initialise orbit graph of $(\langle S, V \rangle)\rho$*]
7: extend $(S)\lambda$ to $(\langle S, V \rangle)\lambda$ [*as described above*]
8: find representatives $(z_1)\lambda, \dots, (z_r)\lambda$ of the s.c.c.s of $(\langle S, V \rangle)\lambda$
9: find the groups S_{z_i} for $i \in \{1, \dots, r\}$ [*Algorithm 4*]
10: **for** $x_j \widehat{y_k} v = \widehat{y_i} \in \widehat{\mathfrak{R}}$ **do** [*j, k are obtained from the Schreier tree for $\widehat{\mathfrak{R}}$, $k < i$*]
11: find $n \in \{1, \dots, r\}$ such that $(z_n)\lambda \sim (\widehat{y_i})\lambda$ (in $(\langle S, V \rangle)\lambda$)
12: find $u \in S$ such that $(z_n)\lambda \cdot u = (\widehat{y_i})\lambda$ [*Algorithm 2*]
13: find $\overline{u} \in U$ such that $(\widehat{y_i})\lambda \cdot \overline{u} = (z_n)\lambda$ and $z_n u \overline{u} = z_n$ [*Assumption* (III)]
14: **for** $y_l \in \mathfrak{R}$ with $(y_l)\rho = (\widehat{y_i})\rho$ and $(y_l)\lambda = (z_n)\lambda$ **do**
15:   **if** $(y_l' \widehat{y_i} \overline{u})\mu_{z_n} \in S_{z_n} = S_{y_l}$ **then** [$\widehat{y_i} \mathscr{R}^{\langle S, V \rangle} y_l$]
16:    $G_{(k)\iota, j} := l$ [*update the orbit graph of $\mathfrak{R}$*]
17:    define $(i)\iota := l$ [*the old index i is the new index l*]
18:    go to line 10
19:   **end if**
20: **end for**
21: $M := M + 1$, $y_M := \widehat{y_i}\overline{u}$, $\mathfrak{R} := \mathfrak{R} \cup \{y_M\}$ [*add $\widehat{y_i}\overline{u}$ to $\mathfrak{R}$*]
22: $V_M := (k)\iota$, $W_M := j$ [*update the Schreier tree of $\mathfrak{R}$*]
23: $K_M := \widehat{K_i}$ [*the index of $(x_j \widehat{y_k})\lambda$ in $(\langle S, V \rangle)\lambda$ from Algorithm 11 applied to S*]
24: $G_{(k)\iota, j} := M$ [*update the orbit graph of $\mathfrak{R}$*]
25: $(i)\iota := M$ [*the old index i is the new index M*]
26: **end for**
27: **return** apply Algorithm 11 to the data structures for $\langle S, V \rangle$ determined so far, and return the output

unary operation ${}^{-1} : x \mapsto x^{-1}$, then Algorithm 6 becomes simpler still. In this case, it is unnecessary to find the s.c.c. of $(x)\rho$. This follows from the observation that we may take $(x)\rho = (x^{-1})\lambda$, which implies that $(S)\rho = (S)\lambda$ and

$$u^{-1} \cdot (x^{-1})\rho = (u^{-1}x^{-1})\rho = (xu)\lambda = (x)\lambda \cdot u.$$

Similar simplifications can be made in Algorithm 8.

The unary operation in the definition of U means that given an inverse subsemigroup S of U and $x \in S$, that we can find x^{-1} without reference to S. Or put differently, the inverse of x is the same in every inverse subsemigroup of U. For example, the symmetric inverse monoid has this property, but the full transformation monoid does not. More precisely, there exist distinct inverse subsemigroups S and T of T_n and $x \in S \cap T$ such that the inverse of x in S is distinct from the inverse of x in T.

As noted above, in a regular subsemigroup S of U, the $\mathscr{R}$-classes are in 1-1 correspondence with the elements of $(S)\rho$ and the $\mathscr{L}$-classes are in 1-1 correspondence with $(S)\lambda$. It follows that the search for $\mathscr{R}$-class representatives in Algorithm 11 is redundant in this case. Hence the $\mathscr{R}$-classes, $\mathscr{L}$-classes, $\mathscr{H}$-classes, size, and elements, of a regular subsemigroup can be determined from $(S)\lambda$ and $(S)\rho$ using Algorithms 2 and 4 alone. The $\mathscr{D}$-classes of a regular subsemigroup S are then in 1-1 correspondence with the s.c.c.s of $(S)\lambda$ (or $(S)\rho$). In the case that U is an inverse semigroup, it suffices to calculate either $(S)\lambda$ or $(S)\rho$, making these computations simpler still. The remaining algorithms in Subsection 5.5 can also be modified to take advantage of these observations, but due to considerations of space, we do not go into the details here.

The simplified algorithms alluded to in this section have been fully implemented in the Semigroups package for GAP; see [28].

6 Examples

In this section we present some examples to illustrate the algorithms from the previous section.

One of the examples is that of a semigroup of partial permutations. Similar to permutations, a partial permutation can be expressed as a union of the components of its action. Any component of the action of a partial permutation f is either a permutation with a single cycle, or a chain $[i\ (i)f\ (i)f^2\ \ldots\ (i)f^r]$ where $i \in \mathrm{dom}(f) \setminus \mathrm{im}(f)$ and $(i)f^r \in \mathrm{im}(f) \setminus \mathrm{dom}(f)$, for some $r > 1$. For the sake of brevity, we will use *disjoint component* notation when writing a specific partial permutation f, i.e. we write f as a juxtaposition of disjoint cycles and chains. For example,

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 4 & - & 2 & 6 & - \end{pmatrix} = [1\ 5\ 6](2\ 4).$$

We include fixed points in the disjoint component notation for a partial permutation f so that it is possible to deduce the domain and image of f from the notation, and so that the notation for f is unique (up to the order of the components, and the order of elements in a cycle); see [25] for further details.

Throughout this section, we will denote by S the subsemigroup of the symmetric inverse monoid on $\{1, \ldots, 9\}$ generated by

$$x_1 = (1\ 4)(2\ 6)(3\ 8)(5)(7)(9), \quad x_2 = (1\ 5\ 4\ 2\ 7\ 6)(3\ 9\ 8) \quad x_3 = (2\ 5\ 6), \quad x_4 = (1\ 3\ 2), \tag{6.1}$$

by T the subsemigroup of the full transformation monoid on $\{1, \ldots, 5\}$ generated by

$$x_1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 2 & 4 & 5 \end{pmatrix}, \quad x_2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}, \quad x_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 3 & 2 & 2 \end{pmatrix}, \tag{6.2}$$

and we will use the notation of Sections 4.1 and 4.2.

Components of the action

Applying Algorithm 1 to S and $\alpha_1 = (x_1)\lambda = \{1, \ldots, 9\}$, we obtain:

$$\begin{array}{rllll} (S)\lambda = \{ & \alpha_1 = \{1, \ldots, 9\}, & \alpha_2 = \{2, 5, 6\}, & \alpha_3 = \{1, 2, 3\}, & \alpha_4 = \{1, 4, 7\}, \\ & \alpha_5 = \{1\}, & \alpha_6 = \{4, 6, 8\}, & \alpha_7 = \{5, 7, 9\}, & \alpha_8 = \{5\}, \\ & \alpha_9 = \varnothing, & \alpha_{10} = \{3\}, & \alpha_{11} = \{4\}, & \alpha_{12} = \{2\}, \\ & \alpha_{13} = \{6\}, & \alpha_{14} = \{8\}, & \alpha_{15} = \{9\}, & \alpha_{16} = \{7\} \quad \}. \end{array}$$

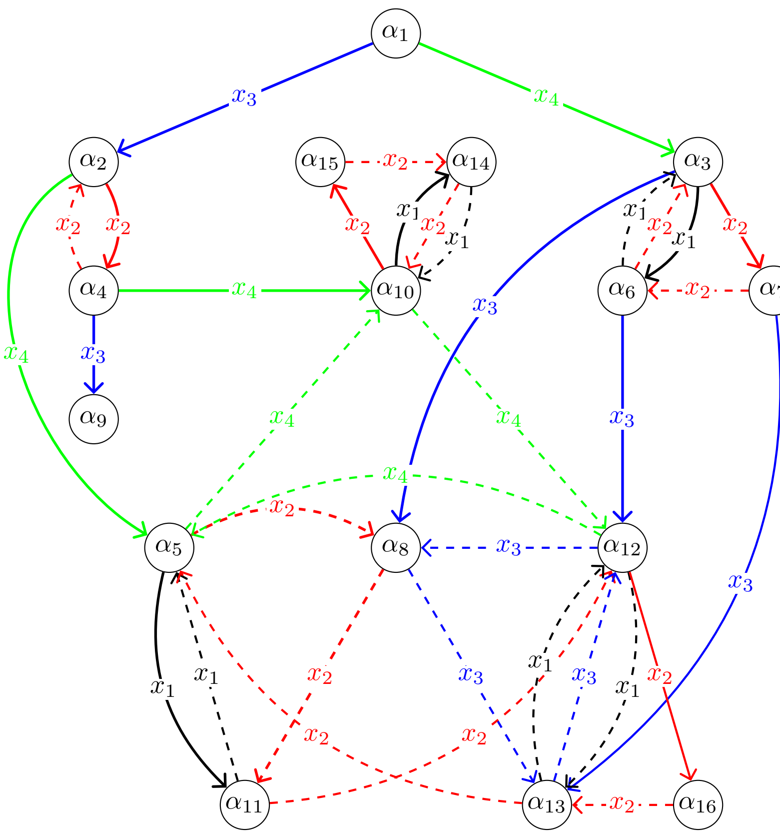

Figure 3: The orbit graph of $(S)\lambda$ with loops and (all but one of the) edges to α_9 omitted, and the Schreier tree indicated by solid edges.

The Schreier tree is:

i	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
v_i	1	1	2	2	3	3	3	4	4	5	6	7	10	10	12
w_i	3	4	2	4	1	2	3	3	4	1	3	3	1	2	2

and the orbit graph is:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$g_{i,1}$	1	2	6	4	11	3	7	8	9	14	5	13	12	10	15	16
$g_{i,2}$	1	4	7	2	8	3	6	11	9	15	12	16	5	10	14	13
$g_{i,3}$	2	2	8	9	9	12	13	13	9	9	9	8	12	9	9	9
$g_{i,4}$	3	5	3	10	10	9	9	9	9	12	9	5	9	9	9	9

.

Diagrams of the Schreier tree and orbit graphs of $(S)\lambda$ can be found in Figure 3.

The strongly connected components of $(S)\lambda$ are:

$$\{\{\alpha_1\}, \{\alpha_2, \alpha_4\}, \{\alpha_3, \alpha_6, \alpha_7\}, \{\alpha_5, \alpha_8, \alpha_{10}, \alpha_{11}, \alpha_{12}, \alpha_{13}, \alpha_{14}, \alpha_{15}, \alpha_{16}\}, \{\alpha_9\}\}.$$

From the Schreier tree, we deduce that

$$\alpha_1 = (x_1)\lambda, \quad \alpha_2 = (x_1x_3)\lambda, \quad \alpha_3 = (x_1x_4)\lambda, \quad \alpha_5 = (x_1x_3x_4)\lambda, \quad \text{and} \quad \alpha_9 = (x_1x_3x_2x_3)\lambda.$$

In this case, we set $\bar{s} = s^{-1}$ in Assumption (III). Using Algorithms 2 and 4, after removing redundant generators, we obtain Schreier generators for the stabilisers of $x_1, x_1x_3, x_1x_4, x_1x_3x_4$, and $x_1x_3x_2x_3$:

$$\begin{array}{lll lll}
S_{x_1} & = & \langle x_1, x_2\rangle \cong D_{12}, & S_{x_1x_3} & = & \langle (2\ 6), (2\ 6\ 5)\rangle \cong \mathrm{Sym}(\{2,5,6\}), \\
S_{x_1x_4} & = & \langle (1\ 3\ 2), (1\ 2)\rangle \cong \mathrm{Sym}(\{1,2,3\}), & S_{x_1x_3x_4} & = & \mathrm{Sym}(\{1\}) \cong \mathbf{1}, \\
S_{x_1x_3x_2x_3} & = & \mathrm{Sym}(\varnothing) \cong \mathbf{1}, & & &
\end{array}$$

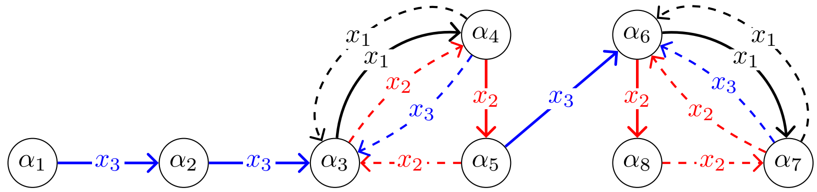

Figure 4: The orbit graph of $(T)\lambda$ with loops omitted, and the Schreier tree indicated by solid edges.

where **1** denotes the trivial group.

Applying Algorithm 1 to T (defined in (6.2)) and $(x_1)\lambda = \{1, 2, 3, 4, 5\}$, we obtain:

$$\begin{array}{llllll}(T)\lambda = \{ & \alpha_1 = \{1,2,3,4,5\}, & \alpha_2 = \{1,2,3\}, & \alpha_3 = \{1,3\}, & \alpha_4 = \{1,2\}, & \\ & \alpha_5 = \{2,3\}, & \alpha_6 = \{3\}, & \alpha_7 = \{2\}, & \alpha_8 = \{1\} & \},\end{array}$$

the Schreier tree is:

i	2	3	4	5	6	7	8
v_i	1	2	3	4	5	6	6
w_i	3	3	1	2	3	1	2

and the orbit graph is:

i	1	2	3	4	5	6	7	8
$g_{i,1}$	1	2	4	3	5	7	6	8
$g_{i,2}$	1	2	4	5	3	8	6	7
$g_{i,3}$	2	3	3	3	6	6	6	8

.

Diagrams of the Schreier tree and orbit graphs of $(T)\lambda$ can be found in Figure 4.

The strongly connected components of $(T)\lambda$ are:

$$\{\{\alpha_1\}, \{\alpha_2\}, \{\alpha_3, \alpha_4, \alpha_5\}, \{\alpha_6, \alpha_7, \alpha_8\}\}.$$

From the Schreier tree for $(T)\lambda$ and Algorithms 2,

$$\alpha_1 = (x_1)\lambda, \quad \alpha_2 = (x_1x_3)\lambda, \quad \alpha_3 = (x_1x_3^2)\lambda, \quad \text{and} \quad \alpha_6 = (x_1x_3^2x_1x_2x_3)\lambda.$$

Using Algorithm 4, after removing redundant generators, we obtain Schreier generators for the stabilisers T_y where $y = x_1, x_1x_3, x_1x_3^2, x_1x_3^2x_1x_2x_3$:

$$\begin{array}{llllll} T_{x_1} & = & \langle x_1, x_2\rangle \cong D_{12}, & T_{x_1x_3} & = & \langle (2\ 3), (1\ 2\ 3)\rangle \cong \mathrm{Sym}(\{1,2,3\}), \\ T_{x_1x_3^2} & = & \langle (1\ 3)\rangle \cong \mathrm{Sym}(\{1,3\}), & T_{x_1x_3^2x_1x_2x_3} & = & \mathrm{Sym}(\{3\}) \cong \mathbf{1}, \end{array}$$

where **1** denotes the trivial group.

Individual Green's classes

Let S be partial permutation semigroup defined in (6.1). The s.c.c. of $(x_1x_3x_4)\lambda$ contains 9 values:

$$\{\alpha_5 = \{1\}, \alpha_8 = \{5\}, \alpha_{10} = \{3\}, \alpha_{11} = \{4\}, \alpha_{12} = \{2\}, \alpha_{13} = \{6\}, \alpha_{14} = \{8\}, \alpha_{15} = \{9\}, \alpha_{16} = \{7\}\},$$

and $|S_{x_1x_3x_4}| = 1$. It follows, by Corollary 3.15(b), that the size of the $\mathscr{R}$-class of $x_1x_3x_4$ in S is $9 \cdot 1 = 9$.

One choice for the Schreier tree (rooted at $\alpha_5 = (x_1x_3x_4)\lambda$) of the s.c.c. of $(x_1x_3x_4)\lambda$ is:

i	8	10	11	12	13	14	15	16
v_i	5	5	5	11	8	10	10	12
w_i	2	4	1	2	3	1	2	2

A diagram of the Schreier tree can be found in Figure 5.

Since $x_1x_3x_4 = [2\ 1]$, using the Schreier tree and Algorithm 5, the elements of $R^S_{x_1x_3x_4}$ are:

$$\begin{array}{llllllllllll} R^S_{x_1x_3x_4} = \{ & x_1x_3x_4 & = & [2\ 1], & x_1x_3x_4^3 & = & (2), & x_1x_3x_4^2 & = & [2\ 3], & \\ & x_1x_3x_4x_1 & = & [2\ 4], & x_1x_3x_4x_2 & = & [2\ 5], & x_1x_3x_4^3x_1 & = & [2\ 6], & \\ & x_1x_3x_4^3x_2 & = & [2\ 7], & x_1x_3x_4^2x_1 & = & [2\ 8], & x_1x_3x_4^2x_2 & = & [2\ 9] & \}, \end{array}$$

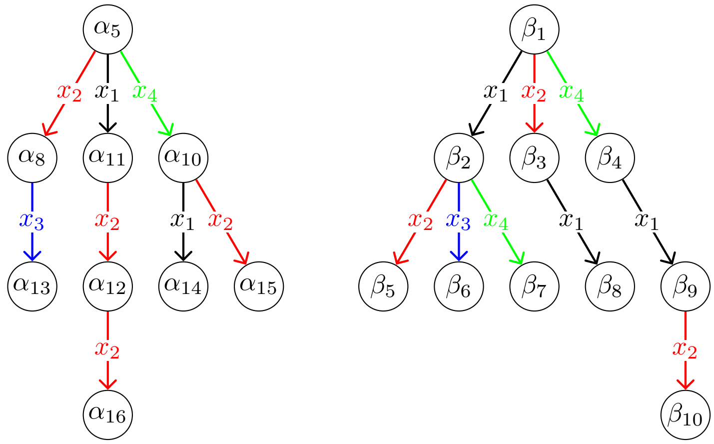

Figure 5: Schreier trees for the strongly connected component of $(x_1x_3x_4)\lambda$ in $(S)\lambda$, and for $(x_1x_3x_4)\rho$ in $(S)\rho$.

Since $x_1x_3x_4^3$ is an idempotent, it also follows that $R^S_{x_1x_3x_3}$ is a regular $\mathscr{R}$-class.

We will calculate the $\mathscr{R}$-classes in $D^S_{x_1x_3x_4}$ using Algorithm 6. Since $(x_1x_3x_4)\rho = \{2\}$, it follows immediately that ${}_{x_1x_3x_4}S = \{\mathrm{id}_{\{2\}}\}$ is trivial. Set $x = x_1x_3x_4$ and $x' = x^{-1} = [1\ 2]$. The embedding $\Psi : {}_xS \longrightarrow U_x$ (from Proposition 3.13(a)) defined by

$$((s)\nu_x)\Psi = (x'sx)\mu_x$$

maps $\mathrm{id}_{\{2\}}$ to $\mathrm{id}_{\{1\}}$. Hence $S_x \cap ({}_xS)\Psi = S_x = \{\mathrm{id}_{\{1\}}\}$. It also follows from Proposition 3.23 that the $\mathscr{R}^S$-class representatives of D^S_x are in 1-1 correspondence with the s.c.c. of $(x)\rho = \{2\}$. Using the left analogue of Algorithm 1, we obtain

$$\begin{array}{rlllll} S\cdot(x)\rho = \{ & \beta_1 = \{2\}, & \beta_2 = \{6\}, & \beta_3 = \{4\}, & \beta_4 = \{3\}, & \beta_5 = \{7\}, \\ & \beta_6 = \{5\}, & \beta_7 = \varnothing, & \beta_8 = \{1\}, & \beta_9 = \{8\}, & \beta_{10} = \{9\} \quad \} \end{array}$$

with Schreier tree:

i	2	3	4	5	6	7	8	9	10
v_i	1	1	1	2	2	2	3	4	9
w_i	1	2	4	2	3	4	1	1	2

A diagram of the Schreier tree can be found in Figure 5. Hence the $\mathscr{R}^S$-class representatives of D^S_x are:

$$\begin{array}{lclllclllcl} x_1x_3x_4 & = & [2\ 1], & x_1^2x_3x_4 & = & [6\ 1], & x_2x_1x_3x_4 & = & [4\ 1], \\ x_4x_1x_3x_4 & = & [3\ 1], & x_2x_1^2x_3x_4 & = & [7\ 1], & x_3x_1^2x_3x_4 & = & [5\ 1], \\ x_1x_2x_1x_3x_4 & = & (1), & x_1x_4x_1x_3x_4 & = & [8\ 1], & x_2x_1x_4x_1x_3x_4 & = & [9\ 1]. \end{array}$$

Since the number of $\mathscr{R}^S$-classes in D^S_x is 9 and each $\mathscr{R}^S$-class has size 9, it follows that $|D^S_x| = 81$.

We will demonstrate how to use Algorithm 7 to check if the partial permutation $y = [1\ 5][2\ 7][3\ 9]$ is $\mathscr{R}^S$-related to either of the generators x_3 and x_4 of S. Since $(y)\rho = \{1, 2, 3\}$ and $(x_3)\rho = \{2, 5, 6\}$, it follows that $(y, x_3) \notin \mathscr{R}^S$. However, $(y)\rho = (x_4)\rho$ and

$$(y)\lambda = \{5, 7, 9\} = \alpha_7 \sim \alpha_3 = \{1, 2, 3\} = (x_4)\lambda.$$

Tracing the Schreier tree of $(S)\lambda$ from α_3 to α_7, we obtain $u = x_2$ such that $(x_4)\lambda \cdot u = (y)\lambda$. It follows that $\overline{u} = x_2^{-1}$ has the property that $(y)\lambda \cdot \overline{u} = (x)\lambda$. Also setting $x_4' = x_4^{-1}$, it follows that $y \in R^S_x$ since

$$(x_4'y\overline{u})\mu_x = (x_4^{-1}yx_2^{-1})\mu_x = (1\ 2\ 3) \in S_{x_4} = S_{x_1x_4} = \mathrm{Sym}(\{1, 2, 3\}).$$

The main algorithm

We now determine the global structure of the transformation semigroup T defined in (6.2). We will do the same thing for the partial permutation semigroup S defined in (6.1) in the next subsection.

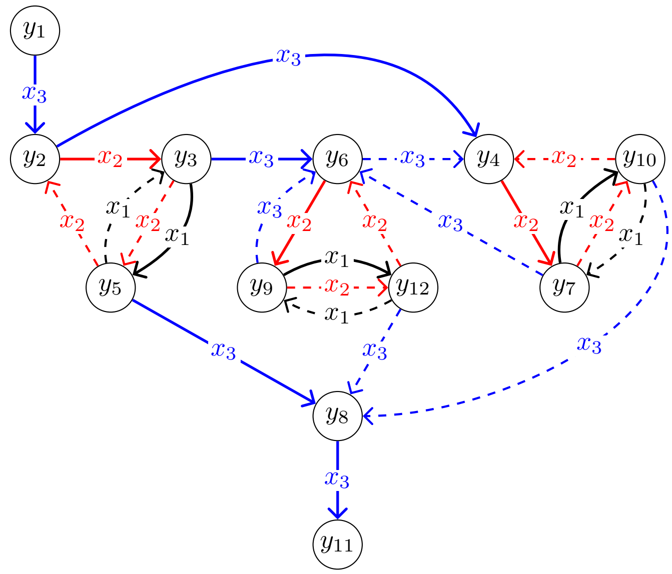

Figure 6: The orbit graph of $(T)\rho$ and $\mathfrak{R}$ with loops omitted, and the Schreier tree indicated by solid edges.

If x is a transformation of degree $n \in \mathbb{N}$, then the kernel $\ker(x)$ of x is a partition of $\{1, \ldots, n\}$. If the classes of $\ker(x)$ are $A_1, A_2, \ldots, A_r$, for some r, then to avoid writing too many brackets we write $\ker(x) = \{A_1 | \cdots | A_r\}$.

Applying Algorithm 11 to T defined in (6.2), we find that the $\mathscr{R}$-class representatives of S are:

$$\begin{array}{llllllllllll}
y_1 & = & 1_T, & y_2 & = & x_3, & y_3 & = & x_2x_3, & y_4 & = & x_3^2, \\
y_5 & = & x_1x_2x_3, & y_6 & = & x_3x_2x_3, & y_7 & = & x_2x_3^2, & y_8 & = & x_3x_1x_2x_3, \\
y_9 & = & (x_2x_3)^2, & y_{10} & = & x_1x_2x_3^2, & y_{11} & = & x_3^2x_1x_2x_3, & y_{12} & = & x_1(x_2x_3)^2,
\end{array}$$

and that

$$(T)\rho = \{ \begin{array}{lllllllll}
(y_1)\rho & = & \{1|2|3|4|5\}, & (y_2)\rho & = & \{1|2,3|4,5\}, & (y_3)\rho & = & \{1,2|3|4,5\}, \\
(y_4)\rho & = & \{1|2,3,4,5\}, & (y_5)\rho & = & \{1,3|2|4,5\}, & (y_6)\rho & = & \{1,4,5|2,3\}, \\
(y_7)\rho & = & \{1,2,4,5|3\}, & (y_8)\rho & = & \{1,2,3|4,5\}, & (y_9)\rho & = & \{1,2|3,4,5\}, \\
(y_{10})\rho & = & \{1,3,4,5|2\}, & (y_{11})\rho & = & \{1,2,3,4,5\}, & (y_{12})\rho & = & \{1,3|2,4,5\}
\end{array} \}.$$

The Schreier tree of the orbit graphs of $\mathfrak{R}$ and $(T)\rho$ are both equal:

i	2	3	4	5	6	7	8	9	10	11	12
v_i	1	2	2	3	3	4	5	6	7	8	9
w_i	3	2	3	1	3	2	3	2	1	3	1

and the orbit graph is:

i	1	2	3	4	5	6	7	8	9	10	11	12
$g_{i,1}$	1	2	5	4	3	6	10	8	12	7	11	9
$g_{i,2}$	1	3	5	7	2	9	10	8	12	4	11	6
$g_{i,3}$	2	4	6	4	8	4	6	11	6	8	11	8

.

A diagrams of the Schreier tree and orbit graph can be found in Figure 6.

It is coincidentally the case that the $\mathscr{R}$-class representatives of T are obtained by left multiplying previous $\mathscr{R}$-class representatives by a generator. In other words, the u and $\overline{u}$ in line 8 and 9 of Algorithm 11 are just the identity of T in this example. Hence the additional parameters returned by Algorithm 11 in this case are $(1, 2, 2, 3, 2, 3, 3, 3, 3, 3, 6, 3)$.

Recall that the strongly connected components of $(T)\lambda$ have representatives: $\alpha_1 = (x_1)\lambda = \{1, \ldots, 5\}$, $\alpha_2 = (x_1x_3)\lambda = \{1, 2, 3\}$, $\alpha_3 = (x_1x_3^2)\lambda = \{1, 3\}$, $\alpha_6 = (x_1x_3^2x_1x_2x_3)\lambda = \{3\}$, and sizes: 1, 1, 3, and 3, respectively. We saw above that $|T_{x_1}| = 12$, $|T_{x_1x_3}| = 6$, $|T_{x_1x_3^2}| = 2$, and $|T_{x_1x_3^2x_1x_2x_3}| = 1$. It follows from Corollary 3.15(b) and (c) that

$$\begin{aligned} |T| &= |R^T_{x_1}| + 3|R^T_{x_1x_3}| + 7|R^T_{x_1x_3^2}| + |R^T_{x_1x_3^2x_1x_2x_3}| \\ &= (1\cdot 1\cdot 12) + (3\cdot 1\cdot 6) + (7\cdot 3\cdot 2) + (1\cdot 3\cdot 1) = 75. \end{aligned}$$

The orbit graph of $\mathfrak{R}$ has 5 strongly connected components:

$$\begin{array}{l} \{y_1 = x_1\}, \qquad \{y_2 = x_3, \quad y_3 = x_2x_3, y_5 = x_1x_2x_3\}, \\ \{y_4 = x_3^2, y_6 = x_3x_2x_3, y_7 = x_2x_3^2, y_9 = (x_2x_3)^2, y_{10} = x_1x_2x_3^2, y_{12} = x_1(x_2x_3)^2\}, \\ \{y_8 = x_3x_1x_2x_3\}, \qquad \{y_{11} = x^3x_1x_2x_3\} \end{array}$$

and so there are five $\mathscr{D}$-classes in T.

Optimizations for inverse semigroups

The semigroup S defined in (6.1) is an inverse semigroup, since the inverses of the generators can be obtained by taking powers. From Section 5.6, it follows that the $\mathscr{R}$-class representatives of S are in 1-1 correspondence with the values in $(S)\lambda$ and that $(S)\rho = (S)\lambda$. Hence the number of $\mathscr{R}$-classes in S is 16, and by tracing the Schreier tree of $(S)\lambda$, the $\mathscr{R}$-class representatives are:

$$\begin{array}{llllllllllll} y_1 &=& x_1, & y_2 &=& x_1x_3, & y_3 &=& x_1x_4, & y_4 &=& x_1x_3x_2, \\ y_5 &=& x_1x_3x_4, & y_6 &=& x_1x_4x_1, & y_7 &=& x_1x_4x_2, & y_8 &=& x_1x_4x_3, \\ y_9 &=& x_1x_3x_2x_3, & y_{10} &=& x_1x_3x_2x_4, & y_{11} &=& x_1x_3x_4x_1, & y_{12} &=& x_1x_4x_1x_3, \\ y_{13} &=& x_1x_4x_2x_3, & y_{14} &=& x_1x_3x_2x_4x_1, & y_{15} &=& x_1x_3x_2x_4x_2, & y_{16} &=& x_1x_4x_1x_3x_2. \end{array}$$

The strongly connected components of $(S)\lambda$ are in 1-1 correspondence with the $\mathscr{D}$-classes of S, and so S has five $\mathscr{D}$-classes. Representatives of $\mathscr{L}$-classes can be obtained by taking the inverses of the $\mathscr{R}$-class representatives $\mathfrak{R}$.

It follows from Corollary 3.15(c) that

$$|S| = (1^2\cdot 12) + (2^2\cdot 6) + (3^2\cdot 6) + (9^2\cdot 1) + 1 = 172.$$

Testing membership

In this subsection, we will use Algorithm 12 to test if the following transformations belong to the semigroup T defined in (6.2):

$$x = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 3 & 1 \end{pmatrix} \qquad \text{and} \qquad y = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 3 & 2 & 2 \end{pmatrix}.$$

Although $(x)\lambda = \{1,2,3\} = \alpha_2 \in (T)\lambda$, $(x)\rho = \{1,5|2|3,4\} \notin (T)\rho$, and so $x \notin T$.

Firstly,

$$(y)\lambda = \{2,3\} = \alpha_5 \in (T)\lambda$$

and so the representative of the s.c.c. of $(y)\lambda$, which we chose above, is α_3. Tracing the Schreier tree for $(T)\lambda$ from α_5 back to α_3, using Algorithm 2, we find $u = x_1x_2$ such that $\alpha_3\cdot u = \alpha_5$. Since x_1, x_2 are permutations, $\overline{u} = x_2^{-1}x_1^{-1}$ has the property that $\alpha_5\cdot\overline{u} = \alpha_3$ and $yu\overline{u} = y$. The $\mathscr{R}$-class representative $y_6 \in \mathfrak{R}$ is the only one such that $(y_6)\lambda = \alpha_3 = (y)\lambda\cdot\overline{u}$ and $(y_6)\rho = \{1,4,5|2,3\} = (y)\rho$. Thus to check that $y \in T$, it suffices to show that the permutation $(y_6'y\overline{u})\mu_x$ belongs to the group $T_{y_6} = T_{x_1x_3^2} = \mathrm{Sym}(\{1,3\})$. We know that $(y_6'y\overline{u})\mu_x$ is a permutation on $\{1,3\}$, and so it must belong to T_{y_6}, and so $y \in S$.

Factorization

In the previous subsection we showed that

$$y = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 3 & 2 & 2 \end{pmatrix}$$

is an element of T defined in (6.2). In this subsection, we will show how to use Algorithm 13 to factorize y as a product of the generators of T. Recall that we will write $y = xsu$, where $x \in \mathfrak{R}$, $u \in S$ is such that $(x)\lambda \cdot u = (y)\lambda$, and $s = x'y\overline{u}$, and that we factorise each of x, s, u separately. From the previous subsection, the chosen $\mathscr{R}$-class representative for y is:

$$x = y_6 = x_3x_2x_3 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 1 & 3 & 3 \end{pmatrix},$$

and one choice for $x' \in T_5$ is:

$$x' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 1 & 4 & 5 \end{pmatrix}.$$

From the previous subsection, $u = x_1x_2$ and $\overline{u} = x_2^{-1}x_1^{-1}$. It follows that

$$s = x'y\overline{u} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 2 & 1 & 4 & 5 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 3 & 2 & 2 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 3 & 4 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 3 & 1 & 1 & 1 \end{pmatrix}$$

and so $(s)\mu_x = s|_{\mathrm{im}(x)} = s|_{\{1,3\}} = (1\ 3)$, which is the only generator of T_x. From Algorithm 4, one choice for s such that $s|_{\mathrm{im}(x)} = (1\ 3)$ is $x_3^2x_1x_2^2$. Hence

$$y = xsu = x_3x_2x_3 \cdot x_3^2x_1x_2^2 \cdot x_1x_2 = x_3x_2x_3^3x_1x_2^2x_1x_2.$$

Note that $x_3x_2x_3x_2^2$ is a minimal length word in the generators that is equal to y.

The $\mathscr{D}$-class structure

We showed above that the partial permutation semigroup S defined in (6.1) has five $\mathscr{D}$-classes D_1, D_2, D_3, D_4, and D_5 with representatives x_1, x_1x_3, x_1x_4, $x_1x_3x_4$, and $x_1x_3x_2x_3$, respectively. The $\mathscr{D}$-class D_1 has only one $\mathscr{R}$-class and one $\mathscr{L}$-class. If we left multiply the unique $\mathscr{R}$-class representative x_1 of D_1 by the generators of S, then we obtain the $\mathscr{R}$-class representatives:

$$x_1^2\mathscr{D}^Sx_1, \quad x_2x_1\mathscr{D}^Sx_1, \quad x_3x_1 = (2\ 5)(6)\mathscr{D}^Sx_1x_3, \quad x_4x_1 = [1\ 8][2\ 4][3\ 6]\mathscr{D}^Sx_1x_4$$

and so $D_2, D_3 \leq_{\mathscr{D}} D_1$. Note that, since S is inverse, we have not performed Algorithm 11, and so we have not previously left multiplied the $\mathscr{R}$-representatives of S by its generators.

Right multiplying the unique $\mathscr{L}$-class representative x_1 of D_1 by the generators of S we obtain:

$$x_1^2\mathscr{D}^Sx_1, \quad x_1x_2\mathscr{D}^Sx_1, \quad x_1x_3 = (2)(5\ 6)\mathscr{D}^Sx_1x_3, \quad x_1x_4 = [4\ 3][6\ 1][8\ 2]\mathscr{D}^Sx_1x_4,$$

which yields no additional information.

Continuing in this way, we obtain the partial order of the $\mathscr{D}$-classes of S. A picture of the egg-box diagrams of the $\mathscr{D}$-classes of S and the partial order of $\mathscr{D}$-classes of S can be seen in Figure 7. An analogous computation can be used to find the partial order of the $\mathscr{D}$-classes of the transformation semigroup T and this is included in Figure 7.

Acknowledgements

The authors would like to thank Wilf Wilson for his careful reading of several early drafts of this paper, and for the many improvements he suggested.

References

[1] Matrix group recognition project, https://www.math.auckland.ac.nz/~henrik/mgrp/, June 2008.

[2] J. Araújo, P. V. Bünau, J. D. Mitchell, and M. Neunhöffer. Computing automorphisms of semigroups. *J. Symbolic Comput.*, 45(3):373–392, 2010.

[3] Martin Beaudry. Membership testing in commutative transformation semigroups. *Inform. and Comput.*, 79(1):84–93, 1988.

[4] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

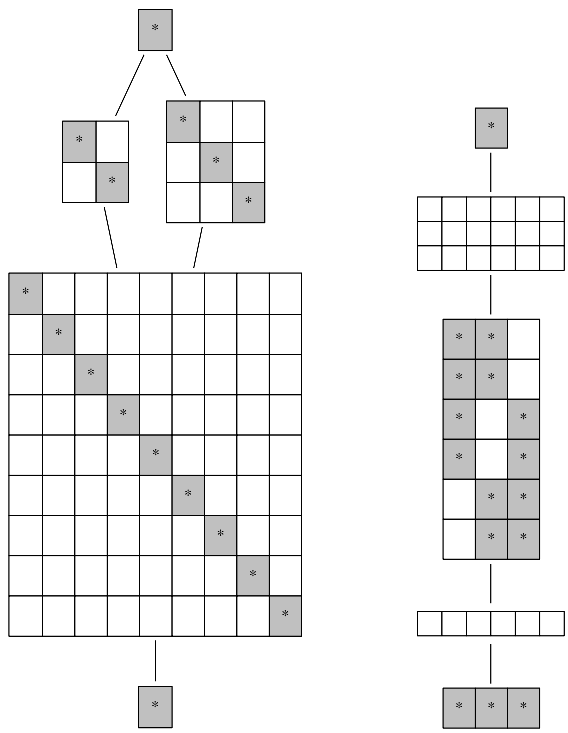

Figure 7: The partial order of the $\mathscr{D}$-classes of S (left) and T (right), group $\mathscr{H}$-classes indicated by shaded boxes.

[5] John J. Cannon. *Computation in finite algebraic structures*. Ph.D. thesis, University of Sydney, 1969.

[6] John J. Cannon. Computing the ideal structure of finite semigroups. *Numer. Math.*, 18:254–266, 1971/72.

[7] J.-M. Champarnaud and G. Hansel. AUTOMATE, a computing package for automata and finite semigroups. *J. Symbolic Comput.*, 12(2):197–220, 1991.

[8] F. G. Cousineau, J. F. Perrot, and J.-M. Rifflet. Apl programs for direct computation of a finite semigroup. In *APL Congress*, volume 73, pages 67–74, Amsterdam, 1973. North-Holland Publishing Co.

[9] J. East and R.D. Gray. Diagram monoids and Graham–Houghton graphs: idempotents and generating sets of ideals. *Preprint*, 2015, `arXiv:1404.2359`.

[10] D. G. Fitz-Gerald. Computing the maximum generalized inverse of a Boolean matrix. *Linear Algebra and Appl.*, 16(3):203–207, 1977.

[11] D.G. FitzGerald and K.W. Lau. On the Partition Monoid and some Related Semigroups. *Bull. Aust. Math. Soc.*, 83(2):273–288, 2011.

[12] George E. Forsythe. SWAC computes 126 distinct semigroups of order 4. *Proc. Amer. Math. Soc.*, 6:443–447, 1955.

[13] Véronique Froidure and Jean-Eric Pin. Algorithms for computing finite semigroups. In *Foundations of computational mathematics (Rio de Janeiro, 1997)*, pages 112–126. Springer, Berlin, 1997.

[14] Harold N. Gabow. Path-based depth-first search for strong and biconnected components. *Information Processing Letters*, 74(34):107 – 114, 2000.

[15] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.7.8*, June 2015.

[16] R. Gray and N. Ruškuc. Generating sets of completely 0-simple semigroups. *Comm. Algebra*, 33(12):4657–4678, 2005.

[17] John M. Howie. *Fundamentals of semigroup theory*, volume 12 of *London Mathematical Society Monographs. New Series*. The Clarendon Press Oxford University Press, New York, 1995. Oxford Science Publications.

[18] H. Jürgensen. Computers in semigroups. *Semigroup Forum*, 15(1):1–20, 1977/78.

[19] Janusz Konieczny. Green's equivalences in finite semigroups of binary relations. *Semigroup Forum*, 48(2):235–252, 1994.

[20] Gerard Lallement and Robert McFadden. On the determination of Green's relations in finite transformation semigroups. *J. Symbolic Comput.*, 10(5):481–498, 1990.

[21] Charles R. Leedham-Green. The computational matrix group project. In *Groups and computation, III (Columbus, OH, 1999)*, volume 8 of *Ohio State Univ. Math. Res. Inst. Publ.*, pages 229–247. de Gruyter, Berlin, 2001.

[22] S. A. Linton, G. Pfeiffer, E. F. Robertson, and N. Ruškuc. *Monoid - GAP 3 package, Version 2.4*. University of St Andrews, September 1997.

[23] S. A. Linton, G. Pfeiffer, E. F. Robertson, and N. Ruškuc. Groups and actions in transformation semigroups. *Math. Z.*, 228(3):435–450, 1998.

[24] S. A. Linton, G. Pfeiffer, E. F. Robertson, and N. Ruškuc. Computing transformation semigroups. *J. Symbolic Comput.*, 33(2):145–162, 2002.

[25] Stephen Lipscomb. *Symmetric inverse semigroups*, volume 46 of *Mathematical Surveys and Monographs*. American Mathematical Society, Providence, RI, 1996.

[26] A. Markov. The impossibility of certain algorithms in the theory of associative systems. II. *Doklady Akad. Nauk SSSR (N.S.)*, 58:353–356, 1947.

[27] D. B. McAlister. *Semigroup for Windows*. University of Northern Illinois, 2006.

[28] J. D. Mitchell et al. *Semigroups - GAP package, Version 2.6*, September 2015.

[29] J. D. Mitchell and Markus Pfeiffer. Computing with semigroups of matrices over finite fields. 2015.

[30] J Müller, F. Noeske, and M. Neunhöffer. *Orb - GAP package, Version 4.6*, May 2013.

[31] T. E. Nordahl and H. E. Scheiblich. Regular $*$-semigroups. *Semigroup Forum*, 16(3):369–377, 1978.

[32] Jan Okniński. *Semigroups of matrices*, volume 6 of *Series in Algebra*. World Scientific Publishing Co. Inc., River Edge, NJ, 1998.

[33] Jean-François Perrot. *Contribution à l'étude des monoïdes syntactiques et de certains groupes associés aux automates finis*. Université de Paris VI, Paris, 1972. Thèse présentée à l'Université Paris VI, Paris, pour l'obtention du Doctorat d'État ès Sciences, Spécialité: Mathématiques.

[34] Jean-Eric Pin. *Semigroupe 2.01: a software for computing finite semigroups*. Laboratoire d'Informatique Algorithmique : Fondements et Applications (LIAFA), CNRS et Université Paris 7, 2.01 edition, 2009.

[35] Emil L. Post. Recursive unsolvability of a problem of Thue. *J. Symbolic Logic*, 12:1–11, 1947.

[36] John Rhodes and Benjamin Steinberg. *The q-theory of finite semigroups*. Springer Monographs in Mathematics. Springer, New York, 2009.

[37] Boris M. Schein. Regular elements of the semigroup of all binary relations. *Semigroup Forum*, 13(2):95–102, 1976/77.

[38] Robert Sedgewick. *Algorithms*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company Advanced Book Program, Reading, MA, 1983.

[39] Ákos Seress. *Permutation group algorithms*, volume 152 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2003.

[40] Imre Simon. Piecewise testable events. In *Automata theory and formal languages (Second GI Conf., Kaiserslautern, 1975)*, pages 214–222. Lecture Notes in Comput. Sci., Vol. 33. Springer, Berlin, 1975.

[41] Charles C. Sims. Computational methods in the study of permutation groups. In *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, pages 169–183. Pergamon, Oxford, 1970.

[42] W. A. Stein et al. *Sage Mathematics Software (Version 5.12)*. The Sage Development Team, 2013. `http://www.sagemath.org`.

[43] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[44] S. Wilcox. Cellularity of Diagram Algebras as Twisted Semigroup Algebras. *J. Algebra*, 309(1):10–31, 2007.

[45] D. Holt with B. Eick and E. O'Brien. *Handbook of computational group theory*. CRC Press, Boca Raton, Ann Arbor, London, Tokyo, 2004.