The Todd-Coxeter Algorithm for Semigroups and Monoids

T. D. H. Coleman, J. D. Mitchell, F. L. Smith, and M. Tsalakou

March 18, 2024

Abstract

In this paper we provide an account of the Todd-Coxeter algorithm for computing congruences on semigroups and monoids. We also give a novel description of an analogue for semigroups of the so-called Felsch strategy from the Todd-Coxeter algorithm for groups.

Contents

1 Introduction ................................................................. 1
2 Prerequisites .............................................................. 2
3 Congruence enumeration .................................................. 4
  3.1 Word graphs .......................................................... 5
  3.2 The definition ......................................................... 5
4 Validity ........................................................................... 7
  4.1 Completeness, determinism, and compatibility ..................... 9
  4.2 The proofs of Theorem 4.1 and Corollary 4.2 ...................... 11
5 Monoids not defined by a presentation ................................. 12
6 The HLT strategy ............................................................ 15
7 The Felsch strategy ......................................................... 16
  7.1 First version ............................................................. 16
  7.2 Second version .......................................................... 18
8 Implementation issues ....................................................... 20
9 Further variants ............................................................. 21
  9.1 Monoids with zero ....................................................... 22
  9.2 Rees congruences ....................................................... 22
  9.3 Stephen’s procedure .................................................... 22
A Extended examples ......................................................... 25
B Performance comparison .................................................. 35
  B.1 Monoids of transformations ......................................... 35
  B.2 Monoids of partitions ................................................ 35
  B.3 Further finitely presented semigroups and monoids ............ 37
  B.4 Finitely presented groups ........................................... 37

1 Introduction

In this article we describe the Todd-Coxeter algorithm for congruence enumeration in semigroups and monoids. The essential purpose of this algorithm is to compute the action of a finitely presented semigroup or monoid on the equivalence
classes of a left, right, or two-sided congruence. Most existing implementations (see, for example, ACE [14], GAP [12], and MAF [38]) and expository accounts (see, for example, [29, p. 351], [31, Sections 4.5, 4.6 and Chapter 5], and [15, Chapter 5]) of the Todd-Coxeter algorithm relate to the enumeration of the cosets of a subgroup of a finitely presented group; or, more precisely, to the production of a permutation representation of the action of the group on the cosets. This was extended to linear representations by Linton [19, 20]. The purpose of this article is to provide an expository, but more or less complete, account of the Todd-Coxeter algorithm for semigroups and monoids.

The Todd-Coxeter algorithm is not a single procedure, but rather an infinite collection of different but related procedures. In the literature for finitely presented groups, examples of procedures in this collection are referred to as coset enumerations; see, for example, [31, Sections 4.5, 4.6 and Chapter 5]. Coset enumerations are also not algorithms, at least by some definitions, in that they might consist of infinitely many steps, and they do not always terminate. In fact, a coset enumeration terminates if and only if the subgroup whose cosets are being enumerated has finite index; for further details see Section 4. This is not to say, however, that the number of steps, or the run time, of a coset enumeration can be predicted in advance; it is relatively straightforward to find examples of finite presentations for the trivial group where the number of steps in a coset enumeration is arbitrarily high. For instance, the group presentation \langle a, b | a^n b^{n+1}, b^n a b^{-1} a^{-1} \rangle defines the trivial group, and is likely that at least \( n \) steps are required in any coset enumeration for this presentation; see [17]. Although this might seem rather negative, nothing more can really be expected. For example, if a coset enumeration for the trivial subgroup of a finitely presented group \( G \) does successfully terminate, then the output can be used to solve the word problem in \( G \). It is well-known by the theorem of Novikov [27] and Boone [3] that the word problem for finitely presented groups is undecidable and so no procedure that solves the word problem, including any coset enumeration, can be expected to terminate in every case. Analogous statements about the undecidability of the word problem hold for finitely presented semigroups and monoids; see [29, Chapter 12].

Congruences are to semigroups and monoids what cosets are to groups, and so we will refer to congruence enumeration for finitely presented semigroups and monoids as the analogue of coset enumeration for groups. We will not consider the special case of enumerating the cosets of a subgroup of a group separately from the more general case of enumerating classes of a congruence of a semigroup.

The first computer implementation of the Todd-Coxeter algorithm is attributed to Haselgrove in 1953 by Leech [18]. Several authors (for example, Neubüser in [25] and Walker in [36]) comment that this may be the first computer implementation of an algorithm for groups, possibly representing the starting point of the field of computational group theory. Neumann [26] adapted the algorithm in [35] to semigroups, and Jura [17] proved that Neumann’s adaptation was valid. One congruence enumeration strategy is described in Ruškuc [30, Chapter 12] as well as variants for computing Rees congruences and minimal ideals. Stephen [32, Chapter 4] also describes a variant of Todd-Coxeter, which can be used to solve the word problem by constructing only part of the action of a finitely presented semigroup or monoid on itself by right multiplication. Stephen’s procedure [32, Section 4.1] is similar to one of the two main strategies for congruence enumeration described in Section 6; see also Section 9.

Versions of the Todd-Coxeter algorithm for semigroups were implemented in FORTRAN by Robertson and Ünlü [28]; in C by Walker [36, 37]; and in GAP by Pfeiffer [12]. The C++ library libsemigroups [24] (by three of the present authors) contains a flexible optimized implementation of the different versions of the Todd-Coxeter algorithm described in this document for semigroups and monoids.

The paper is organised as follows: some mathematical prerequisites from the theory of monoids are given in Section 2; in Section 3 we define the infinite family of procedures for enumerating a right congruence on a monoid; the validity of congruence enumeration is shown in Section 4; in Section 5 we discuss how congruence enumeration can be used to compute congruences of a monoid that is not given \( a \text{ priori} \) by a finite presentation; in Section 6 and Section 7 we describe the two main strategies for congruence enumeration; in Section 8 we discuss some issues related to the implementation of congruence enumeration; and, finally, in Section 9 we present some variants: for enumerating congruences of monoids with zero elements, Rees congruences, and Stephen’s procedure [32, Section 4.1].

In Appendix A we give a number of extended examples, and in Appendix B there is a comparison of the performance of the implementations in libsemigroups [24] of various strategies when applied to a number of group, monoid, or semigroup presentations. The authors have also implemented basic versions of the algorithms described in this paper in python3 as a concrete alternative to the description given here; see [23].

## 2 Prerequisites

In this section, we provide some required mathematical prerequisites. The contents of this section are well-known in the theory of semigroups; for further background information we refer the reader to [16].

If \( \mu \) is an equivalence relation on a set \( X \), then we denote by \( x/\mu \) the equivalence class of \( x \) in \( \mu \) and by \( X/\mu \) the set \( \{ x/\mu : x \in X \} \) of equivalence classes. The least equivalence relation on \( X \) with respect to containment is
\[
\begin{array}{c}
X \times S \rightarrow Y \times S \\
\downarrow \downarrow \\
X \lambda \rightarrow Y
\end{array}
\]

Figure 2.1: A commutative diagram illustrating a homomorphism of actions where \(X \times S \rightarrow Y \times S\) is the function defined by \((x, s) \mapsto ((x)\lambda, s)\).

\[\Delta_X = \{ (x, x) \in X \times X : x \in X \};\] we refer to this as the **trivial equivalence** on \(X\).

If \(S\) is a semigroup and \(R \subseteq S \times S\), then there is an **(two-sided) elementary sequence with respect to \(R\)** between \(x, y \in S\) if \(x = y\) or \(x = s_1, s_2, \ldots, s_m = y\), for some \(m \geq 2\), where \(s_i = p_i u q_i, s_{i+1} = p_i v q_i\), with \(p_i, q_i \in S^1\) and \((u_i, v_i) \in R\) for all \(i\). If \(p_i = \varepsilon\) for all \(i\) in an elementary sequence, then we refer to the sequence as a **right elementary sequence**; left elementary sequences are defined analogously. We define \(R^\# \subseteq S \times S\) so that \((x, y) \in R^\#\) if and only if there is an elementary sequence between \(x\) and \(y\) with respect to \(R\). Note that \(R^\#\) is the least congruence on the semigroup \(S\) containing \(R\); see [16, Section 1.5]. The least right congruence containing a subset of \(S \times S\) is defined analogously, using right elementary sequences rather than two-sided elementary sequences; we will not reserve any special notation for such least right congruences. Least left congruences are defined dually.

Throughout this article we write functions to the right of their arguments, and compose from left to right. Let \(S\) be a semigroup and let \(X\) be a set. A function \(\Psi : X \times S \rightarrow X\) is a **right action** of \(S\) on \(X\) if \((x, s)\Psi = (x, st)\Psi\) for all \(x \in X\) and for all \(s, t \in S\). If in addition \(S\) has an identity element \(e\), we require \((x, e)\Psi = x\) for all \(x \in X\) also.

If \(S\) is a semigroup, then we may adjoin an identity to \(S\) (if necessary) so that \(S\) is a monoid. We denote \(S\) with an adjoined identity \(1_S\) by \(S^1\). If \(\Psi : X \times S \rightarrow X\) is a right action of a semigroup \(S\) on a set \(X\), then \(\Psi^1 : X \times S^1 \rightarrow X\) defined by \((x, s)\Psi^1 = (x, s)\Psi\) for all \(x \in X\) and \(s \in S\), and \((x, 1_S)\Psi^1 = x\) for all \(x \in X\), is a right action of \(S^1\) on \(X\) also.

For the sake of brevity, we will write \(x \cdot s\) instead of \((x, s)\Psi\), and we will say that \(S\) acts on \(X\) on the right. The **kernel** of a function \(f : X \rightarrow Y\), where \(X\) and \(Y\) are any sets, is the equivalence relation

\[
\ker(f) = \{ (x, y) \in X \times X : (x)f = (y)f \}.
\]

If \(X\) is any set, then \(X^X\) denotes the set of all functions from \(X\) to \(X\). Endowed with the operation of function composition, \(X^X\) is a monoid, called the **full transformation monoid** on \(X\). A right action \(\Psi\) of a semigroup \(S\) on a set \(X\) induces a homomorphism \(\Psi^1 : S \rightarrow X^X\) defined by \((s)\phi : x \mapsto x \cdot s\) for all \(s \in S\) and all \(x \in X\).

The **kernel** of a right action \(\Psi\) of a semigroup \(S\) on a set \(X\) is the kernel of the function \(\Psi^1 : S \rightarrow X^X\),

\[
\ker(\Psi) = \{ (s, t) : (s)\Psi^1 = (t)\Psi^1 \} = \{ (s, t) \in S \times S : (x, s)\Psi = (x, t)\Psi, \text{ for all } x \in X \}.
\]

It is straightforward to verify that the kernel of a homomorphism, and the kernel of a right action of a semigroup \(S\), is a congruence on \(S\).

If \(S\) acts on the sets \(X\) and \(Y\) on the right, then we say that \(\lambda : X \rightarrow Y\) is a **homomorphism of right actions** if \((x \cdot s)\lambda = (x\lambda) \cdot s\) for all \(x \in X\) and \(s \in S^1\) (see Fig. 2.1 for a diagram). An **isomorphism** of right actions is a bijective homomorphism of right actions.

If \(A\) is a set, then a **word** over \(A\) is a finite sequence \((a_1, \ldots, a_n)\) where \(a_1, \ldots, a_n \in A\) and \(n \geq 0\); say that the **length** of the word is \(n\). We will write \(a_1 \cdots a_n\) rather than \((a_1, \ldots, a_n)\), and will refer to \(A\) as an **alphabet** and \(a \in A\) as a **letter**. If \(A\) is any set, then the **free semigroup on** \(A\) is the set \(A^+\) consisting of all words over \(A\) of length at least 1 with operation given by the juxtaposition of words. The **free monoid** on \(A\) is the set \(A^+ \cup \{ \varepsilon \}\) with the operation again the juxtaposition of words and where \(\varepsilon\) denotes the unique word of length 0, the so-called **empty word**; the free monoid on \(A\) is denoted \(A^*\).

The next proposition might be viewed as an analogue of the Third Isomorphism Theorem for semigroups ([16, Theorem 1.5.4]) in the context of actions. More precisely, Proposition 2.1 will allow us to replace the action of a free monoid on a quotient of a quotient by an action on a single quotient. We require the following definition. Let \(S\) be a semigroup, let \(\rho\) be a congruence of \(S\), and let \(\sigma\) be a right congruence of \(S\) such that \(\rho \subseteq \sigma\). The set \(\sigma/\rho\) defined as

\[
\sigma/\rho = \{ (x/\rho, y/\rho) \in S/\rho \times S/\rho : (x, y) \in \sigma \}
\]

is a right congruence on \(S/\rho\).

---

1This is an abuse of notation since \(\ker(\Psi)\) might mean its kernel as an action or as a function. However, the reader may be reassured to know that if \(\Psi\) is an action, we will never use \(\ker(\Psi)\) to mean its kernel as a function.
Proposition 2.1. Let $S$ be a semigroup, let $\rho$ be a congruence of $S$, and let $\sigma$ be a right congruence of $S$ such that $\rho \subseteq \sigma$. The right actions of $S$ on $(S/\rho)/(\sigma/\rho)$ and $S/\sigma$ defined by
\[
(u/\rho)/(\sigma/\rho) \cdot v = (uv/\rho)/(\sigma/\rho) \quad \text{and} \quad u/\sigma \cdot v = uv/\sigma \quad \text{for all} \ u, v \in S
\]
are isomorphic.

The proof of Proposition 2.1 is similar to the proof of [16, Theorem 1.5.4]. An analogue of Proposition 2.1 can be formulated for left congruences, but we do not require this explicitly; see Proposition 3.1, and the surrounding text, for further details.

We also require the following result, the proof of which is routine, and hence omitted.

Proposition 2.2. Let $S$ be a semigroup, let $X$ be a set, and let $\Psi : X \times S \rightarrow X$ be a right action. If $\mu \subseteq \ker(\Psi)$ is a congruence on $S$, then $\overline{\Psi} : X \times S/\mu \rightarrow X$ defined by
\[
(x, s/\mu)\overline{\Psi} = (x, s)\overline{\Psi}
\]
for all $x \in X$ and $s \in S$, is a right action that is isomorphic to $\Psi$.

3 Congruence enumeration

In this section we define what we mean by a congruence enumeration for a semigroup or monoid, and establish some further notational conventions. Congruence enumeration, as described in this section, will provide the general context for the more explicit algorithms discussed in Section 6 and Section 7; this section is based, at least in spirit, on [31, Section 5.1] and is also influenced by [32, 33].

The purpose of a congruence enumeration is to determine, in some sense, the structure of a finitely presented monoid; or more generally, a congruence on such a monoid. For the sake of simplicity, we will assume throughout that $P = \langle A | R \rangle$ is a monoid presentation defining a monoid $M$, where $A$ is some totally ordered finite alphabet, and $R$ is a (possibly empty) finite subset of $A^* \times A^*$. Additionally, we suppose that $S$ is a finite subset of $A^* \times A^*$. We write $R^\#$ to be the least two-sided congruence on $A^*$ containing $R$, and denote by $\rho$ the least right congruence on $A^*$ containing $R^\#$ and $S$. In this notation, the monoid $M$ defined by the presentation $P$ is $A^*/R^\#$. If the number of congruence classes of $\rho$ is finite, then the output of a congruence enumeration is a description of the natural right action of $A^*/\rho$ on the congruence classes of $\rho$. In particular, the output yields the number of such classes, can be used to determine whether or not two words in $A^*$ belong to the same class, and provides a homomorphism from $A^*$ to the full transformation monoid $(A^*/\rho)^{\langle A^*/\rho \rangle}$ on the set of congruence classes of $\rho$. Since it is undecidable whether a finitely presented monoid is finite or not [22] (see also [4, Remark 4]), an upper bound for the number of steps required for a congruence enumeration for $\langle A | R \rangle$ to terminate cannot be computed as a function of $|A|$ and $|R|$. If the number of congruence classes of $\rho$ is infinite, then the enumeration will not terminate. If the set $S$ is empty, then a congruence enumeration, if it terminates, gives us a description of the monoid $A^*/R^\#$ together with the natural right action of $A^*$ on $A^*/R^\#$.

Perhaps it is more natural to want to determine a right congruence of the monoid $M$ defined by the presentation $P$ rather than on the free monoid $A^*$. However, by Proposition 2.1, to determine the right congruence on $M = A^*/R^\#$ generated by
\[
\{ (s/R^\#, t/R^\#) : (s, t) \in S \}
\]
it suffices to determine the right congruence $\rho$ on $A^*$ containing $R^\#$ and the set $S$.

It is only for the sake of convenience that we have chosen to consider finitely presented monoids, rather than finitely presented semigroups. To apply the algorithms described in this article to a semigroup that is not a monoid, simply adjoin an identity, perform the algorithm, and disregard the adjoined identity in the output.

The choice of “right” rather than “left” in the previous paragraphs was arbitrary. If $w = a_1a_2 \cdots a_n \in A^*$, then we denote the reversed word $a_n \cdots a_2a_1$ by $w^\dagger$. If $W \subseteq A^* \times A^*$, then we denote by $W^\dagger$ the set $\{ (w^\dagger, v^\dagger) : (u, v) \in W \}$.

Proposition 3.1. Let $A$ be an alphabet, let $R, S \subseteq A^* \times A^*$, let $R^\#$ be the least congruence on $A^*$ containing $R$, and let $\rho$ be the least left congruence on $A^*$ containing $R^\#$ and $S$. Then $\rho^\dagger$ is the least right congruence on $A^*$ containing $R^\#^\dagger$ and $S^\dagger$.

It follows from Proposition 3.1 that any algorithm for computing right congruences can be used for left congruences, by simply reversing every word.

The inputs of a congruence enumeration are the finite alphabet $A$, the finite set of relations $R \subseteq A^* \times A^*$, the finite set $S \subseteq A^* \times A^*$, and a certain type of digraph that is defined in the next section.
3.1 Word graphs

One of the central components of the proofs presented in Section 4 is that of a word graph, which is used in a natural representation of equivalence relations on a free monoid. Let $A$ be any alphabet and let $\Gamma = (N, E)$ be a digraph with non-empty finite set of nodes $N \subseteq \mathbb{N}$ with $0 \in N$ and edges $E \subseteq N \times A \times N$. Then, following [32, 33], we refer to $\Gamma$ as a word graph. The word graph $\Gamma = (\{0\}, \emptyset)$ is the trivial word graph.

If $(a, a, \beta) \in E$ is an edge in a word graph $\Gamma$, then $\alpha$ is the source, $a$ is the label, and $\beta$ is the target of $(a, a, \beta)$. An edge $(a, a, \beta)$ is said to be incident to its source $\alpha$ and target $\beta$.

A word graph $\Gamma$ is deterministic if for every node $\alpha$ and every letter $a \in A$ there is at most one edge of the form $(a, a, \beta)$ in $\Gamma$. A word graph $\Gamma$ is complete if for every node $\alpha$ and every letter $a \in A$ there is at least one edge incident to $\alpha$ labelled by $a$ in $\Gamma$.

If $\alpha, \beta \in N$, then an $(\alpha, \beta)$-path is a sequence of edges $(a_1, a_2, \ldots, a_n) \in E$ where $a_1 = \alpha$ and $a_{n+1} = \beta$ and $a_1, \ldots, a_n \in A$; $\alpha$ is the source of the path; the word $a_1 \cdots a_n \in A^*$ labels the path; $\beta$ is the target of the path; and the length of the path is $n$. If $\alpha, \beta \in N$ and there is an $(\alpha, \beta)$-path in $\Gamma$, then we say that $\beta$ is reachable from $\alpha$.

If $\Gamma = (N, E)$ is a word graph and $\mathfrak{P}(A^* \times A^*)$ denotes the power set of $A^* \times A^*$, then the path relation of $\Gamma$ is the function $\pi_\Gamma : N \rightarrow \mathfrak{P}(A^* \times A^*)$ defined by $(u, v) \in (\alpha)\pi_\Gamma$ if there exists a node $\beta$ such that $u$ and $v$ both label $(\alpha, \beta)$-paths in $\Gamma$.

If $\Gamma$ is a word graph and $\alpha$ is a node in $\Gamma$, then $(\alpha)\pi_\Gamma$ is reflexive and symmetric, and $(\alpha)\pi_\Gamma$ is transitive for all $\alpha$ if and only if $\Gamma$ is deterministic. In particular, $(\alpha)\pi_\Gamma$ is an equivalence relation for all $\alpha$ if and only if $\Gamma$ is deterministic. If $R \subseteq A^* \times A^*$, $\Gamma$ is a word graph, and $\pi_\Gamma$ is the path relation of $\Gamma$, then we say that $\Gamma$ is compatible with $R$ if $R \subseteq (\alpha)\pi_\Gamma$ for every node $\alpha$ in $\Gamma$.

Suppose that $\Gamma = (N, E)$ is a word graph with nodes $N$ and edges $E \subseteq N \times A \times N$, and $\kappa \subseteq N \times N$ is an equivalence relation. The quotient $\Gamma/\kappa$ of $\Gamma$ by $\kappa$ is the word graph with nodes $\{\min \alpha/\kappa : \alpha \in N\}$, and edges $(\min \alpha/\kappa, a, \min \beta/\kappa)$ whenever $(\alpha, a, \beta) \in E$. It is routine to verify that if $\Gamma = (N, E)$ is a word graph, $\kappa$ is any equivalence relation on $N$, and $\kappa'$ is any equivalence relation on the set of nodes of $\Gamma/\kappa$, then $(\Gamma/\kappa)/(\kappa') = \Gamma/\kappa''$ where $\kappa''$ is the least equivalence on $N$ containing $\kappa$ and every $(\alpha, \beta) \in N \times N$ where $\min \alpha/\kappa$, $\min \beta/\kappa \in \kappa'$. If $\Gamma_1 = (N_1, E_1)$ and $\Gamma_2 = (N_2, E_2)$ are word graphs over the same alphabet $A$, then $\phi : N_1 \rightarrow N_2$ is a homomorphism if $(\alpha, a, \beta) \in E_1$ implies $((\alpha)\phi, a, (\beta)\phi) \in E_2$; and we write $\phi : \Gamma_1 \rightarrow \Gamma_2$. An isomorphism of word graphs $\Gamma_1$ and $\Gamma_2$ is a bijection $\phi : \Gamma_1 \rightarrow \Gamma_2$ such that both $\phi$ and $\phi^{-1}$ are homomorphisms. If $\kappa$ is any equivalence relation on a word graph $\Gamma = (N, E)$, then the function $\phi : \Gamma \rightarrow \Gamma/\kappa$ defined by $(\alpha)\phi = \alpha/\kappa$ is a homomorphism. Conversely, if $\phi : \Gamma_1 \rightarrow \Gamma_2$ is a homomorphism, then $((\alpha)\phi) \in E_2$ if and only if $\phi : \Gamma_1 \rightarrow \Gamma_2$ is an isomorphism to the quotient $\Gamma_1/\ker(\phi)$. If $(a_1, a_2, \ldots, a_n, a_{n+1}) \in E_1$ is a path in $\Gamma_1$ and $\phi : \Gamma_1 \rightarrow \Gamma_2$ is a homomorphism, then, by definition, $(\phi(a_1), \phi(a_2), \ldots, (\phi(a_n), (\phi(a_{n+1})) \in E_2$ is a path in $\Gamma_2$ with the same label $a_1 \cdots a_n \in A^*$. In this way, we say that homomorphisms of word graphs preserve paths and labels of paths. This leads to the following straightforward lemma.

**Lemma 3.2.** Let $\Gamma_1 = (N_1, E_1)$ and $\Gamma_2 = (N_2, E_2)$ be word graphs, and let $\pi_{\Gamma_1} : N_1 \rightarrow \mathfrak{P}(A^* \times A^*)$ and $\pi_{\Gamma_2} : N_2 \rightarrow \mathfrak{P}(A^* \times A^*)$ be the path relations of $\Gamma_1$ and $\Gamma_2$, respectively. If $\phi : \Gamma_1 \rightarrow \Gamma_2$ is a homomorphism and $\alpha \in N_1$ is arbitrary, then $(\alpha)\pi_{\Gamma_1} \subseteq ((\alpha)\phi)\pi_{\Gamma_2}$.

Before giving the definition of a congruence enumeration, we highlight that many of the accounts of the Todd-Coxeter algorithm (see, for instance, [15, 25, 31]) are not formulated in terms of digraphs, but rather as a table whose rows are labelled by a set $C$ of non-negative integers, and columns are labelled by the generating set $A$. If $\Gamma = (N, E)$ is a deterministic word graph and $f : C \rightarrow N$ is a bijection such that $(0)f = 0$, then the value in the row labelled $c$ and column labelled $a$ is $f^{-1}$ of the target of the unique edge in $\Gamma$ with source $(c)f$ and label $a$. According to Neubüser [25], until the 1950s, congruence enumeration was often performed by hand, and, in this context, using tables is more straightforward than using graphs. On the other hand, the language of word graphs provide a more accessible means of discussing congruence enumeration in theory.

3.2 The definition

Recall that we suppose throughout that $|A|R$ for some $R \subseteq A^* \times A^*$ is a finite monoid presentation. Additionally, we will suppose throughout that $A$ is a totally ordered alphabet. A congruence enumeration is a sequence of the following steps TC1, TC2, and TC3 where the input to the $i$-th step (where $i \in N$) is $(\Gamma_i, \kappa_i)$ for some word graph $\Gamma_i$ with a totally ordered set of nodes $N_i$ and set of edges $E_i$, and some equivalence relation $\kappa_i \subseteq N_i \times N_i$.

**TC1 (define a new node).** If $\alpha$ is a node in $\Gamma_i$ and there is no edge in $\Gamma_i$ labelled by $a \in A$ with source $\alpha$, then we define $\Gamma_{i+1}$ to be the word graph obtained from $\Gamma_i$ by adding the new node $\beta := 1 + \max \bigcup_{j \leq i} N_j$ and the edge $(\alpha, a, \beta)$. We define $\kappa_{i+1} := \kappa_i \cup \{(\beta, \beta)\}$.
We define $\Gamma \subseteq \mathcal{P}(A^* \times A^*)$ as the least right congruence on $A^*$ containing both $R^#$ and $S$, and that $\Gamma = (N_1, E_1)$ is a word graph with path relation $\pi_{\Gamma_1} : N_1 \to \mathcal{B}(A^* \times A^*)$ such that $(0)\pi_{\Gamma_1} \subseteq \rho$ and $\kappa_1 = \Delta_{N_1}$. Then a congruence enumeration for $\rho$ with input $(\Gamma_1, \kappa_1)$ consists of:

(a) For every $(u, v) \in S$, by repeatedly applying TC1 (if necessary), add edges to $\Gamma_1$ so that it contains paths labelled by $u$ and $v$ both with source $0$.

(b) Apply TC2 to 0 and every $(u, v) \in S$. 

TC2 (follow the paths defined by a relation). Suppose that $\alpha \in N_1$ and $(u, v) \in R$ where $u = u_1a$ and $v = v_1b$ for some $u_1, v_1 \in A^*$ and $a, b \in A$.

(a) If $u$ and $v_1$ label paths from $\alpha$ to some nodes $\beta, \gamma \in N_1$ in $\Gamma_i$, respectively, but $\gamma$ is not the source of any edge labelled by $b$, then we set $\Gamma_{i+1}$ to be the word graph obtained from $\Gamma_i$ by adding the edge $(\gamma, b, \beta)$ and we define $\kappa_{i+1} := \kappa_i$.

(b) The dual of (a) where there are paths labelled by $u_1$ and $v$ to nodes $\beta$ and $\gamma$, respectively, but $\beta$ is not the source of any edge labelled by $a$.

(c) If $u$ and $v$ label paths from $w$ to some nodes $\beta$ and $\gamma$, respectively, and $\beta \neq \gamma$, then we define $\Gamma_{i+1} := \Gamma_i$ and $\kappa_{i+1}$ to be the least equivalence containing $\kappa_i$ and $(\beta, \gamma)$.

Note that conditions (a), (b), and (c) are mutually exclusive, and it may be the case that none of them hold.

TC3 (process coincidences or a determination). We define $\Gamma_{i+1}$ to be the quotient of $\Gamma_i$ by $\kappa_i$ and define $\kappa_{i+1}$ to be the least equivalence on $N_{i+1}$ containing every $(\beta, \gamma)$, $\beta \neq \gamma$, for which there exist $\alpha \in N_{i+1}$ and $a \in A$ such that $(\alpha, a, \beta, \gamma) \in E_{i+1}$. Recall that the quotient $\Gamma_i/\kappa_i$ of $\Gamma_i$ by $\kappa_i$ is the word graph with nodes $\{\min \alpha/\kappa_i : \alpha \in N_i\}$ and hence after an application of TC3 each node in $\Gamma_{i+1}$ is set to be equal to the minimum of the set of nodes in its equivalence class in $\kappa_i$.

There are only finitely many possible quotients of any word graph. Hence if TC3 is applied repeatedly, then after finitely many iterations the output $\kappa_{i+1}$ will equal $\Delta_{N_{i+1}}$, and $\Gamma_{i+1}$ and $\Gamma_i$ will be equal.

If $w$ labels an $(\alpha, \beta)$-path $P$ in some $\Gamma_i$, then neither TC1 nor TC2 changes any of the edges belonging to $P$. Hence if $\Gamma_{i+1}$ is obtained from $\Gamma_i$ by applying TC1 or TC2, then $w$ labels an $(\alpha, \beta)$-path in $\Gamma_{i+1}$ also. If $\Gamma_{i+1}$ is obtained from $\Gamma_i$ by applying TC3, then $\Gamma_{i+1}$ is a homomorphic image of $\Gamma_i$. As already noted, homomorphisms preserve paths, and so $w$ labels a path in $\Gamma_{i+1}$ also.

We can now formally define a congruence enumeration.

**Definition 3.3.** [Congruence enumeration.] Suppose that $A$ is a finite alphabet, that $R, S \subseteq A^* \times A^*$ are finite, that $\rho$ is the least right congruence on $A^*$ containing both $R^#$ and $S$, and that $\Gamma_1 = (N_1, E_1)$ is a word graph with path relation $\pi_{\Gamma_1} : N_1 \to \mathcal{B}(A^* \times A^*)$ such that $(0)\pi_{\Gamma_1} \subseteq \rho$ and $\kappa_1 = \Delta_{N_1}$. Then a congruence enumeration for $\rho$ with input $(\Gamma_1, \kappa_1)$ consists of:

(a) For every $(u, v) \in S$, by repeatedly applying TC1 (if necessary), add edges to $\Gamma_1$ so that it contains paths labelled by $u$ and $v$ both with source 0.

(b) Apply TC2 to 0 and every $(u, v) \in S$. 

Figure 3.1: A diagram representing TC2(a), solid lines correspond to paths in $\Gamma_1$ and the dashed edge is the one defined in TC2(a).
If \((\Gamma_m, \kappa_m)\) is the output of steps (a) and (b), then the enumeration is concluded by performing any sequence of applications of \(TC1, TC2,\) and \(TC3\) such that the following conditions hold for \(\Gamma_i = (N_i, E_i)\) for every \(i \in \mathbb{N}, i \geq m:\)

(c) If \(\alpha \in N_i\) and there is no edge incident to \(\alpha\) with label \(a \in A\), then there exists \(j \geq i\) such that either: \(\alpha\) is no longer a node in \(\Gamma_j\); or there is an edge incident to \(\alpha\) with label \(a\) in \(\Gamma_j\).

(d) If \(\alpha \in N_i\) and \((u, v) \in R\), then there exists \(j \geq i\) such that either: \(\alpha\) is no longer a node in \(\Gamma_j\) or \((u, v) \in (\alpha)\pi_{\Gamma_j}\).

(e) If \(\kappa_i \neq \Delta_{N_i}\) for some \(i\), then there exists \(j \geq i\) such that \(\kappa_j = \Delta_{N_j}\).

The initial value of the word graph \(\Gamma_1\) that forms the input to a congruence enumeration is usually either the trivial word graph or, if \(M\) is finite, the right Cayley graph of the monoid \(M\) defined by the presentation \((A| R)\); see Section 5 for more details.

A congruence enumeration **terminates** if the output \((\Gamma_i, \kappa_i)\) has the property that applying any of \(TC1, TC2,\) or \(TC3\) to \((\Gamma_i, \kappa_i)\) results in no changes to the output, i.e. \((\Gamma_{i+1}, \kappa_{i+1}) = (\Gamma_i, \kappa_i)\). It is straightforward to verify that a congruence enumeration terminates at step \(i\) if and only if \(\Gamma_i\) is complete, compatible with \(R \cup S\), and deterministic.

For any given finite monoid presentation, there is a wide range of choices for the order in which steps \(TC1, TC2,\) and \(TC3\) are performed, and to which nodes, generators, and relations they are applied. We will examine two specific strategies for enumerating congruences for an arbitrary finite monoid presentation in more detail in Section 6 and Section 7.

4 Validity

In this section we address the validity of congruence enumeration as defined in Section 3; we will continue to use the notation established therein.

The main results in this section are the following.

**Theorem 4.1.** Let \(A\) be a finite alphabet, let \(R \subseteq A^* \times A^*\) be a finite set, and let \(R^\#\) be the least two-sided congruence on \(A^*\) containing \(R\). If \(S \subseteq A^* \times A^*\) is any finite set, and \(\rho\) is the least right congruence on \(A^*\) containing \(R^\#\) and \(S\), then the following hold:

(a) If a congruence enumeration for \(\rho\) terminates with output word graph \(\Gamma = (N, E)\), then \(A^*/\rho\) is finite and the function \(\phi : N \times A^* \rightarrow N\), defined by \((\alpha, w)\phi = \beta\) whenever \(w\) labels an \((\alpha, \beta)\)-path in \(\Gamma\), is a right action that is isomorphic to the natural action of \(A^*/\rho\) by right multiplication.

(b) If \(A^*/\rho\) is finite, then any congruence enumeration for \(\rho\) terminates.

**Corollary 4.2.** Let \(A\) be a finite alphabet, let \(R \subseteq A^* \times A^*\) be a finite set, and let \(R^\#\) be the least two-sided congruence on \(A^*\) containing \(R\). Then the following hold:

(a) If a congruence enumeration for \(R^\#\) terminates with output word graph \(\Gamma = (N, E)\), then \(A^*/R^\#\) is finite and the function \(\phi : N \times A^* \rightarrow N\), defined by \((\alpha, w)\phi = \beta\) whenever \(w\) labels an \((\alpha, \beta)\)-path in \(\Gamma\) is a right action that is isomorphic to the (faithful) natural action of \(A^*/R^\#\) on itself by right multiplication.

(b) If \(A^*/R^\#\) is finite, then any congruence enumeration for \(R^\#\) terminates.

We will prove Theorem 4.1 and Corollary 4.2 in Section 4.2. We start by showing that the steps \(TC1, TC2,\) and \(TC3\) preserve certain properties of word graphs in Proposition 4.4. In Section 4.1, we show that every congruence enumeration eventually stabilises and is eventually compatible with \(R\); and in Section 4.2 we give the proofs of Theorem 4.1 and Corollary 4.2.

We will make repeated use, without reference, to the following straightforward lemma.

**Lemma 4.3.** If any of \(TC1, TC2,\) or \(TC3\) is applied to \((\Gamma_i, \kappa_i)\) where every node in \(\Gamma_i\) is reachable from \(0\), then every node in the output \(\Gamma_{i+1}\) satisfies is reachable from \(0\) also.

The next proposition also plays a crucial role in the proof of Theorem 4.1 and Corollary 4.2.

**Proposition 4.4.** If \(TC1, TC2,\) or \(TC3\) is applied to \((\Gamma_i, \kappa_i)\) where \(\Gamma_i = (N_i, E_i)\) is a word graph such that \((0)\pi_{\Gamma_i} \subseteq \rho,\) and \((0)\pi_{\Gamma_i}/\kappa_i \subseteq \rho,\) then the output \((\Gamma_{i+1}, \kappa_{i+1})\) satisfies \((0)\pi_{\Gamma_{i+1}} \subseteq (0)\pi_{\Gamma_{i+1}}/\rho,\) and \((0)\pi_{\Gamma_{i+1}}/\kappa_i \subseteq (0)\pi_{\Gamma_{i+1}}/\kappa_{i+1} \subseteq \rho.\)

The proof of Proposition 4.4 is split into two parts due to commonalities in the proofs rather than the statements. The two cases are: \(TC1, TC2(a), TC2(b);\) and \(TC2(c), TC3.\)
Lemma 4.5. Proposition 4.4 holds when TC1, TC2(a), or TC2(b) is applied.

Proof. Since (a) and (b) of TC2 are symmetric, it suffices to prove the proposition when TC1 or TC2(a) is applied. In TC1, α is a node in Γ_i and there is no edge incident to α labelled by α. In this case, Γ_{i+1} is obtained from Γ_i by adding the single node β := 1 + min\{\sum_{j \leq i} N_j\} and the single edge (α, α, β), and κ_{i+1} := κ_i \cup \{ (β, β) \}. In TC2(a), there exists (u, v) ∈ R, b ∈ A, and v_1 ∈ A^* such that v = v_1 b, and there exist nodes α, β, and γ in Γ_i such that u and v_1 label (α, β)- and (α, γ)-paths in Γ_i, respectively. In this case, Γ_{i+1} is obtained from Γ_i by adding the single edge (γ, b, β) and κ_{i+1} := κ_i.

Hence if either TC1 or TC2(a) is applied, then the identity map N_i \ → \ N_{i+1} is a homomorphism from Γ_i to Γ_{i+1} and so the identity map N_i/κ_i \ → \ N_{i+1}/κ_{i+1} is also a homomorphism from Γ_i/κ_i to Γ_{i+1}/κ_{i+1}. Hence, by Lemma 3.2, (0)π_{Γ_{i+1}} \ ⊆ \ (0)π_{Γ_{i+1}/κ_{i+1}} and (0)π_{Γ_{i+1}/κ_i} \ ⊆ \ (0)π_{Γ_i/κ_i}.

To show that (0)ϕ_{Γ_{i+1}} \ ⊆ \ ρ we consider the cases when TC1 and TC2(a) are applied separately.

TC1: Suppose that (u, v) ∈ (0)π_{Γ_{i+1}} \ \setminus\ \{0\}π_{Γ_i} is arbitrary. Then there are paths in Γ_{i+1} from 0 to some node γ labelled by u and v but there are no such paths in Γ_i. Since β is not a node in Γ_i, β is not the target of any path in Γ_i. It follows that the edge (α, α, β) must occur at least once in both paths. But β is the source of no edges in Γ_{i+1}, and so (α, α, β) occurs once, and it must be the last edge, in both paths. Hence u = u_1 a and v = v_1 a where u_1, v_1 ∈ A^* label (0, α)-paths in Γ_i. Therefore (u_1, v_1) ∈ ρ by assumption and since ρ is a right congruence, (u, v) = (u_1 a, v_1 a) ∈ ρ as well. Hence (0)π_{Γ_{i+1}} \ ⊆ \ ρ.

TC2(a): We proceed by induction on the total number k of occurrences of the edge (γ, b, β) (defined at the start of the proof) in any pair of paths in Γ_{i+1} with source 0 and the same target node. The inductive hypothesis is: if (x, y) ∈ (0)π_{Γ_{i+1}}, X and Y are paths with source 0 labelled by x and y, respectively, and the total number of occurrences of the edge (γ, b, β) in X is strictly less than k, then (x, y) ∈ ρ. The base case is when k = 0 and, in this case, (x, y) ∈ (0)π_{Γ_i} \ ⊆ \ ρ, as required.

Suppose that k ≥ 1, that x, y ∈ A^* are any words labelling (0, δ)-paths X and Y, respectively, for some node δ, and that (γ, b, β) occurs k times in X and Y. Without loss of generality we can assume that (γ, b, β) occurs at least once in X. If X consists of the edges ε_1,ε_2,...,ε_{i+1} and j \in\{1,\ldots,j\} is the largest value such that ε_j = (γ, b, β), then we may write x = x_1 b x_2 where x_1 labels the (0, γ)-path (ε_1,...,ε_{j−1}) and x_2 labels the (β, δ)-path (ε_{j+1},...,ε_i). If w' labels any path from 0 to α in Γ_i, then w'v_1 labels an (0, γ)-path in Γ_i. It follows that (x_1, w'v_1) ∈ (0)π_{Γ_{i+1}}. The number of occurrences of (γ, b, β) in (ε_1,...,ε_{j−1}) is strictly less than k, and the number in the path (in Γ_i) labelled by w'v_1 is 1. Hence, by the inductive hypothesis, (x_1, w'v_1) ∈ ρ and so (x, w'v_2x_2) = (x_1 b x_2, w_1 b x_2) ∈ ρ.

The word w' labels a (0, δ)-path, u labels a (α, β)-path, and x_2 labels a (β, δ)-path. The first two of these paths belong to Γ_i by assumption, and the third does also, by the maximality of j. Hence w'ux_2 labels a (0, δ)-path in Γ_i, which therefore contains no occurrences of the edge (γ, b, β). The word y also labels the (0, δ)-path Y in Γ_{i+1} and this path contains at most k − 1 occurrences of (γ, b, β), since X and Y together contained k occurrences, and X contained at least 1 occurrence of (γ, b, β). Therefore, by induction, (w'ux_2, y) ∈ ρ. Finally, since (u, v) ∈ R \subseteq\ ρ, (w'ux_2, w'vx_2) ∈ R^\# \subseteq \ ρ also. Hence (x, w'vx_2), (w'ux_2, w'vx_2), (w'ux_2, y) ∈ ρ and so by transitivity, (x, y) ∈ ρ.

We conclude the proof by showing that (0)π_{Γ_{i+1}/κ_i} \ ⊆ \ ρ when either TC1 or TC2(a) is applied. Suppose that (u, v) ∈ (0)π_{Γ_{i+1}/κ_i}. If (u, v) ∈ (0)π_{Γ_{i+1}}, then (u, v) ∈ ρ, as required. If (u, v) \not\in (0)π_{Γ_{i+1}}, then u and v label (0, ζ)- and (0, η)-paths in Γ_{i+1}, respectively, for some nodes ζ ≠ η. Recall that κ_{i+1} = κ_i \cup \{ (β, β) \} if TC1 is applied, and κ_{i+1} = κ_i if TC2(a) is applied. In either case, (ζ, η) ∈ κ_{i+1} and ζ ≠ η implies that (ζ, η) ∈ κ_i. But ζ and η are nodes in Γ_i and hence they are reachable from 0 in Γ_i by Lemma 4.3. In other words, there exist u', v' ∈ A^* labelling (0, ζ)- and (0, η)-paths in Γ_i, respectively. Hence (u, u'), (v', v) ∈ (0)π_{Γ_{i+1}} \ ⊆ \ ρ and, by construction, (u', v') ∈ (0)π_{Γ_i/κ_i} \subseteq \ ρ. Thus, by transitivity, (u, v) ∈ ρ, as required.

Lemma 4.6. Proposition 4.4 holds when TC2(c) or TC3 is applied.

Proof. If TC2(c) is applied, then there exist a node α in Γ_i and (u, v) ∈ R such that u and v label (α, β)- and (α, γ)-paths in Γ_i for some distinct nodes β and γ in Γ_i. In this case, Γ_{i+1} = Γ_i and κ_{i+1} is the least equivalence relation containing κ_i and (β, γ). If TC2(c) is applied, then Γ_{i+1} = Γ_i/κ_i and κ_{i+1} is the least equivalence on N_{i+1} containing every (δ, ζ), δ ≠ ζ, for which there exist η ∈ N_i+1 and a ∈ A such that (η, a, δ), (η, a, ζ) ∈ E_{i+1}.

In either case, there is a homomorphism from Γ_{i+1} to Γ_i and so Lemma 3.2 implies that (0)π_{Γ_{i+1}} \subseteq (0)π_{Γ_i/κ_i}. If TC2(c) is applied, then the identity map is a homomorphism from Γ_i to Γ_{i+1}. If TC3 is applied, then Γ_{i+1} = Γ_i/κ_i is a homomorphic image of Γ_i also. Hence, in either case, (0)π_{Γ_i} \subseteq (0)π_{Γ_{i+1}/κ_{i+1}} by Lemma 3.2.

Hence, in both cases, it suffices to show that (0)π_{Γ_i/κ_i} \subseteq (0)π_{Γ_{i+1}/κ_{i+1}} \subseteq \ ρ. We consider the cases when TC2(c) and TC3 are applied separately.
Corollary 4.8. If \( G_i \) is complete at some step \( i \) of a congruence enumeration, then the enumeration terminates at some step \( j \geq i \).

The next lemma shows that, roughly speaking, if the word graph at some step of a congruence enumeration is non-deterministic, then at some later step it is deterministic.

Lemma 4.9. If \( w \in A^* \) labels a path in \( G_i \) starting at 0 for some \( i \in N \), then there exists \( j \geq i \) such that \( w \) labels a unique path in \( G_j \) starting at 0.

Proof. We proceed by induction on the length of the word \( w \). If \( w = \varepsilon \), then \( w \) labels a unique path in every \( G_i \) starting (and ending) at 0.

Suppose that every word of length at most \( n - 1 \) labelling a path starting at 0 in \( G_i \) labels a unique path, and let \( w \in A^* \) be any word of length \( n \geq 1 \), labelling a path in \( G_i \) starting at 0. If \( w = w_1a \) for some \( w_1 \in A^* \) and \( a \in A \), then \( w_1 \) labels a unique path in \( G_i \) starting at 0 by induction. Suppose that \( a \) is the target of the unique path labelled \( w_1 \). By Lemma 4.7, we may assume without loss of generality that \( a \) is a node in \( G_i \) for every \( j \geq i \). If there exist edges \((\alpha, a, \beta_1), \ldots, (\alpha, a, \beta_r)\) in \( G_i \), then at most one of these edges was created by an application of TC1 or TC2. So, if \( r > 1 \), then \((\beta_1, \beta_2), \ldots, (\beta_{r-1}, \beta_r) \in \kappa_i \).
By part (c) of the definition of a congruence enumeration (Definition 3.3), there exists \( j > i \) such that \( \kappa_j = \Delta_N \) and so at step \( j \), there is only one edge incident to \( \alpha \) labelled \( a \).

\[ \square \]

Next we show that every congruence enumeration is eventually compatible with \( \rho \).

**Lemma 4.10.** If \((u, v) \in R^\#\), then there exists \( K \in \mathbb{N} \) such that \((u, v) \in (0)\pi_{\Gamma_i} \) for all \( i \geq K \).

**Proof.** If \((u, v) \in R^\#\), then there exists an elementary sequence \( u = w_1, w_2, \ldots, w_k = v \) in \( A^* \) with respect to \( R \). If \((w_j, w_{j+1}) \in (0)\pi_{\Gamma_i} \), for every \( j \in \{1, \ldots, k - 1\} \), by transitivity, \((w_1, w_k) = (u, v) \in (0)\pi_{\Gamma_i} \) for a large enough \( i \). Hence it suffices to show that \((w_j, w_{j+1}) \in (0)\pi_{\Gamma_j} \) for all \( j \) and for sufficiently large \( i \).

Suppose that \( j \in \{1, \ldots, k - 1\} \) is arbitrary, and that \( w_j = pxq \) and \( w_{j+1} = pyq \) where \( p, q \in A^* \) and \((x, y) \in R \). From part (c) of the definition of a congruence enumeration (Definition 3.3), \( p \) labels a path with source \( 0 \) in \( \Gamma_j \) for all \( j \geq i \) for some sufficiently large \( i \). We may choose \( i \) large enough so that the target node \( \alpha \) of this path is a node in every \( \Gamma_j \) for \( j \geq i \). By Definition 3.3(d), for sufficiently large \( i \), \((x, y) \in (\alpha)\pi_{\Gamma_i} \) and \((px, py) \in (0)\pi_{\Gamma_i} \). Again if \( i \) is large enough, there is a path labelled \( q \) from the target of the path from \( 0 \) labelled by \( px \) (or \( py \)) and so \((w_j, w_{j+1}) = (pxq, pyq) \in (0)\pi_{\Gamma_i} \). \( \square \)

**Corollary 4.11.** If \((u, v) \in \rho \), then there exists \( K \in \mathbb{N} \) such that \((u, v) \in (0)\pi_{\Gamma_i} \) for all \( i \geq K \).

**Proof.** Recall that \( \rho \) is the least right congruence containing \( R^\# \) and \( S \) (as defined in Theorem 4.1). If \((u, v) \in \rho \), then there exists a right elementary sequence \( u = w_1, w_2, \ldots, w_k = v \) such that \( w_j = xq \) and \( w_{j+1} = yq \) for some \((x, y) \in R^\# \cup S \) and some \( q \in A^* \). Hence it suffices to show that if \((x, y) \in R^\# \cup S \) and \( q \in A^* \), then, for sufficiently large \( i \), \((xq, yq) \in (0)\pi_{\Gamma_i} \).

If \((\Gamma_m, \kappa_m) \) is the output of applying steps (a) and (b) of the congruence enumeration (Definition 3.3), then \( S \subseteq (\pi_{\Gamma_m}/\kappa_m)(0) \). Proposition 4.4 implies that \( (\pi_{\Gamma_m}/\kappa_m)(0) \subseteq (\pi_{\Gamma_i}/\kappa_i)(0) \) for all \( i \geq m \). By Definition 3.3(e), \( \text{TC3} \) is eventually applied to \((\Gamma_{i-1}, \kappa_{i-1}) \) for some \( i > m \). Hence \( S \subseteq (0)\pi_{\Gamma_{i-1}/\kappa_{i-1}} = (0)\pi_{\Gamma_i} \), since \( \Gamma_i = \Gamma_{i-1}/\kappa_{i-1} \), for large enough \( i \).

Suppose that \( q \in A^* \) and \((x, y) \in R^\# \cup S \) are arbitrary. If \((x, y) \in R^\# \) then \((x, y) \in (0)\pi_{\Gamma_i} \) for large enough \( i \) by Lemma 4.10. If \((x, y) \in S \), then it follows from the previous paragraph that \((x, y) \in (0)\pi_{\Gamma_i} \) for large enough \( i \). As in the proof of Lemma 4.10, for sufficiently large \( i \), \( xq \) and \( yq \) label paths in \( \Gamma_i \), and so \((xq, yq) \in (0)\pi_{\Gamma_i} \). \( \square \)

**Corollary 4.12.** Suppose that \((u, v) \in A^* \) are arbitrary. Then \((u, v) \in \rho \) if and only if there exists \( K \in \mathbb{N} \) such that \((u, v) \in (0)\pi_{\Gamma_i} \) for all \( i \geq K \).

**Proof.** (\( \Rightarrow \)) Assume that \((u, v) \in \rho \). By Lemma 4.10, \((u, v) \in (0)\pi_{\Gamma_K} \) for large enough \( K \). But \((0)\pi_{\Gamma_i} \subseteq (0)\pi_{\Gamma_{i+1}} \) for all \( i \), and so \((u, v) \in (0)\pi_{\Gamma_i} \) for all \( i \geq K \).

(\( \Leftarrow \)) By assumption, \( \Gamma_1 \) is a word graph with \((0)\pi_{\Gamma_1} \subseteq \rho \). By Proposition 4.4, \((0)\pi_{\Gamma_i} \subseteq \rho \) for all \( i \in \mathbb{N} \). Hence if \((u, v) \in (0)\pi_{\Gamma_i} \) for some \( i \), then \((u, v) \in \rho \). \( \square \)

We also obtain the following stronger result for an enumeration of a two-sided congruence.

**Corollary 4.13.** Suppose that \((u, v) \in A^* \) are arbitrary. Then the following are equivalent:

(i) \((u, v) \in R^\# \);

(ii) there exists \( K \in \mathbb{N} \) such that \((u, v) \in (\alpha)\pi_{\Gamma_i} \) for all nodes \( \alpha \) in \( \Gamma_i \) and for all \( i \geq K \);

(iii) there exists \( K \in \mathbb{N} \) such that \((u, v) \in (0)\pi_{\Gamma_i} \) for all \( i \geq K \).

**Proof.** (i) \( \Rightarrow \) (ii). If \((u, v) \in R^\# \) and \( w \in A^* \) is arbitrary, then \((wu, vw) \in R^\# \) and so, by Lemma 4.10, \((wu, vw) \in (0)\pi_{\Gamma_i} \) for sufficiently large \( i \). In particular, for large enough \( i \), \( wu \) and \( vw \) both label \((0, \beta)\)-paths for some node \( \beta \). If \( \alpha \) is the target of the path starting at \( 0 \) labelled by \( w \), then \( u \) and \( v \) both label a \((\alpha, \beta)\)-paths in \( \Gamma_i \) and so \((u, v) \in (\alpha)\pi_{\Gamma_i} \) for \( i \) large enough.

(ii) \( \Rightarrow \) (iii). This follows immediately since 0 is a node in every word graph.

(iii) \( \Rightarrow \) (i). Follows by applying Corollary 4.12 to \( \rho = R^\# \). \( \square \)
4.2 The proofs of Theorem 4.1 and Corollary 4.2

In this section, we prove Theorem 4.1 and Corollary 4.2.

**Theorem 4.1.** Let $A$ be a finite alphabet, let $R \subseteq A^* \times A^*$ be a finite set, and let $R^\#$ be the least two-sided congruence on $A^*$ containing $R$. If $S \subseteq A^* \times A^*$ is any finite set, and $\rho$ is the least right congruence on $A^*$ containing $R^\#$ and $S$, then the following hold:

(a) If a congruence enumeration for $\rho$ terminates with output word graph $\Gamma = (N, E)$, then $A^*/\rho$ is finite and the function $\phi : N \times A^* \rightarrow N$, defined by $(\alpha, w)\phi = \beta$ whenever $w$ labels an $(\alpha, \beta)$-path in $\Gamma$, is a right action that is isomorphic to the natural action of $A^*$ on $A^*/\rho$ by right multiplication.

(b) If $A^*/\rho$ is finite, then any congruence enumeration for $\rho$ terminates.

**Proof.** We start the proof by showing that there is always a bijection between a certain set of nodes in the word graphs $\Gamma_i$ (defined in a moment) and the classes of $\rho$. Let $M$ be the subset of $\bigcup_{i \in \mathbb{N}} N_i$ so that $\alpha \in M$ if $\alpha \in N_i$ for all large enough $i$. By Lemma 4.7, for every $\alpha \in M$ there exists $w_\alpha \in A^*$ such that $w_\alpha$ labels a path from 0 to $\alpha$ in every $\Gamma_i$ for sufficiently large $i$. We define $f : M \rightarrow A^*/\rho$ by

$$f(\alpha) = w_\alpha/\rho.$$  \hspace{1cm} (4.1)

We will show that $f$ is a bijection.

If $\alpha, \beta \in M$ are such that $(\alpha)f = (\beta)f$, then $(w_\alpha, w_\beta) \in \rho$ and so $(w_\alpha, w_\beta) \in (0)\pi_{\Gamma_i}$ for large enough $i$ by Corollary 4.12. In particular, $w_\alpha$ and $w_\beta$ both label paths from 0 to the same node in every $\Gamma_i$ for large enough $i$. By Lemma 4.7, we can also choose $i$ large enough so that the target $\gamma$ of these paths is a node in $\Gamma_j$ for all $j \geq i$. But $w_\alpha$ and $w_\beta$ also label $(0, \alpha)$- and $(0, \beta)$-paths in every $\Gamma_i$ for sufficiently large $i$. By Lemma 4.9, there exists $j \geq i$ such that $w_\alpha$ labels a unique path starting at 0, and so $\alpha = \gamma$. Similarly, there exists $k \geq j$ such that $w_\beta$ labels a unique path from 0, and so $\beta = \gamma$ also. In particular, $\alpha = \beta$ and so $f$ is injective.

For surjectivity, let $u \in A^*$ be arbitrary. Then $u$ labels a path in $\Gamma_i$ for large enough $i$. Since every congruence enumeration stabilises, we can choose $i$ large enough so that every edge in the path labelled by $u$ belongs to every $\Gamma_j$ for $j \geq i$. In particular, $u$ labels a $(0, \alpha)$-path in $\Gamma_i$ for some $\alpha \in M$. But $w_\alpha$ also labels a $(0, \alpha)$-path in $\Gamma_i$, and so $(u, w_\alpha) \in (0)\pi_{\Gamma_i}$. Hence, again by Corollary 4.12, for large enough $i$, $(u, w_\alpha) \in \rho$ and so $(\alpha)f = w_\alpha/\rho = u/\rho$, and $f$ is surjective.

(a). Suppose that $(\Gamma, \kappa)$ is the output of the enumeration where $\Gamma = (N, E)$ and that $\pi : N \rightarrow \mathcal{P}(A^* \times A^*)$ is the path relation of $\Gamma$. Since the congruence enumeration process has terminated, $\Gamma$ is finite and its set of nodes $N$ coincides with the set $M$ defined at the start of the proof. Hence $A^*/\rho$ is finite, since $f : N \rightarrow A^*/\rho$ defined in (4.1) is a bijection.

Since any application of TC3 results in no changes to $\Gamma$, it follows that $\kappa = \Delta_N \nu$. Hence $\Gamma$ is deterministic, and so every $w \in A^*$ labels exactly one path starting at every $u \in N$. In particular, $\phi$ (as defined in the statement) is a well-defined function. If $w_1, w_2 \in A^*$ label $(\alpha_1, \alpha_2)$- and $(\alpha_2, \alpha_3)$-paths in $\Gamma$, respectively, then certainly $w_1w_2$ labels a $(\alpha_1, \alpha_3)$-path, and so $\phi$ is an action.

We will show that $f : N \rightarrow A^*/\rho$ defined in (4.1) is an isomorphism of the actions $\phi$ and the natural action $\Phi : A^*/\rho \times A^* \rightarrow A^*/\rho$ defined by $(u/\rho, v)\Phi = uv/\rho$. We define $g : N \times A^* \rightarrow (A^*/\rho) \times A^*$ by $(\alpha, v)g = ((\alpha)f, v) = (w_\alpha/\rho, v)$ for all $(\alpha, v) \in N \times A^*$. To show that $f$ is a homomorphism of actions, we must prove that the diagram in Fig. 4.1 commutes. Suppose that $(\alpha, v) \in N \times A^*$. Then

$$(\alpha, v)\Phi = (w_\alpha/\rho, v)\Phi = w_\alpha v/\rho$$

and if $v$ labels a $(\alpha, \beta)$-path in $\Gamma$, then

$$(\alpha, v)\phi = (\beta)f = w_\beta/\rho.$$  \hspace{1cm} (4.2)

But $w_\beta$ and $w_\alpha v$ both label $(0, \beta)$-paths in $\Gamma$ and so $(w_\alpha v, w_\beta) \in (0)\pi$. By Corollary 4.12, $(0)\pi \subseteq \rho$ and so $(w_\alpha v, w_\beta) \in \rho$. Hence $((\alpha, v)\Phi = ((\alpha, v)\phi)f$ and so $f$ is homomorphism of the actions $\phi$ and $\Phi$. 
(b). If $A^*/\rho$ is finite, then the set $M$ of nodes defined at the start of the proof is also finite. Since every congruence enumeration stabilises, it follows that for every $\alpha \in M$, there exists $K \in \mathbb{N}$ such that $(\alpha, a, \beta) \in E_i$ for all $i \geq K$. It follows that $\beta \in M$. Since $A$ and $M$ are finite, there exists $K \in \mathbb{N}$ such that $(\alpha, a, \beta) \in E_i$ for all $\alpha \in M$, all $a \in A$, and all $i \geq K$. It follows that none of TC1, TC2, nor TC3 results in any changes to $\Gamma_i$ or $\kappa_i$, $i \geq K$, and so the enumeration terminates.

We conclude this section by proving Corollary 4.2.

**Corollary 4.2.** Let $A$ be a finite alphabet, let $R \subseteq A^* \times A^*$ be a finite set, and let $R^\# \subseteq A^* \times A^*$ be the least two-sided congruence on $A^*$ containing $R$. Then the following hold:

(a) If a congruence enumeration for $R^\#$ terminates with output word graph $\Gamma = (N, E)$, then $A^*/R^\#$ is finite and the function $\phi : N \times A^* \to N$ defined by $(\alpha, w)\phi = \beta$ whenever $w$ labels an $(\alpha, \beta)$-path in $\Gamma$ is a right action that is isomorphic to the (faithful) natural action of $A^*/R^\#$ on itself by right multiplication.

(b) If $A^*/R^\#$ is finite, then any congruence enumeration for $R^\#$ terminates.

**Proof.** If $S = \emptyset$ in Theorem 4.1, then the least right congruence $\rho$ containing $S$ and $R^\#$ is just $R^\#$. By Theorem 4.1, the action $\phi : N \times A^* \to N$ is isomorphic to the natural action of $A^*$ on $A^*/R^\#$ by right multiplication. By Corollary 4.13, the kernel of $\phi$ is $R^\#$, and it is routine to verify that the kernel of the action of $A^*$ on $A^*/R^\#$ by right multiplication is also $R^\#$. Hence, by Proposition 2.2, the action $\phi$ of $A^*$ on $N$ and the induced action of $A^*/R^\#$ on $N$ are isomorphic.

Similarly, the action of $A^*$ on $A^*/R^\#$ is isomorphic to the action of $A^*/R^\#$ on $A^*/R^\#$ by right multiplication. Thus the actions of $A^*/R^\#$ on $N$ and $A^*/R^\#$ on $A^*/R^\#$ are isomorphic also. The latter action is faithful since $A^*/R^\#$ is a monoid. Hence the action of $A^*/R^\#$ on $N$ is faithful also, and there is nothing further to prove.

5 Monoids not defined by a presentation

Suppose that the monoid $M$ defined by the presentation $\langle A \mid R \rangle$ is finite and that $\phi : A^* \to M$ is the unique surjective homomorphism extending the inclusion of $A$ in $M$. Then $\ker(\phi) = R^\#$ and $(a)\phi = a$ for every $a \in A$. If $M = \{m_1, m_2, \ldots, m_{|M|}\}$, then the right Cayley graph of $M$ with respect to $A$ is the word graph $\Gamma$ with nodes $N = \{0, \ldots, |M| - 1\}$ and edges $\langle (\alpha, a, \beta) \rangle$ for all $\alpha, \beta \in N$ and every $a \in A$ such that $m_\alpha a = m_\beta$. The right Cayley graph $\Gamma$ is complete and deterministic, and so if $\pi : N \to \mathfrak{P}(A^* \times A^*)$ is the path relation on $\Gamma$, then $(\alpha)\pi = R^\#$ for all $\alpha \in N$.

**Example 5.1.** Suppose that $M$ is the monoid generated by the matrices

$$a = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad c = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

over the boolean semiring $\mathbb{B} = \{0, 1\}$ with addition defined by

$$0 + 1 = 1 + 0 = 1 + 1 = 1, \quad 0 + 0 = 0$$

and multiplication defined as usual for the real numbers 0 and 1. The right Cayley graph of $M$ with respect to its generating set $\{a, b, c\}$ is shown in Fig. 5.1; see also Table 5.1.

Suppose that we want to determine a right congruence on a finite monoid $M$ generated by a non-abstract set of generators $A$ — such as transformations, partial permutations, or matrices over a semiring. Of course, every finite monoid $M$ is finitely presented, and so congruence enumeration can be applied to any presentation for $M$. It is possible that a presentation for $M$ is already known, or we can compute a presentation for $M$; for example, by using the Froidure-Pin Algorithm (see [11]).

It is also possible, instead of starting with the trivial word graph, to start a congruence enumeration with the right Cayley graph $\Gamma_1 = (N_1, E_1)$ of $M$ with respect to $A$ as the input to the process. As mentioned above, $(0)\pi_1 = R^\#$ and

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| \(\varepsilon\) | a | b | c | \(a^2\) | ab | ba | bc | bab |

Table 5.1: A word labelling a path from 0 to each node in the right Cayley graph on the monoid $M$ from Example 5.1; see Fig. 5.1.
Figure 5.1: The right Cayley graph of the monoid $M$ from Example 5.1 with respect to the generating set $\{a, b, c\}$; see Table 5.1 for a representative word for each node. Purple arrows correspond to edges labelled $a$, gray labelled $b$, and pink labelled $c$.

so $(\Gamma_1, \Delta_{N_1})$ is a valid input to a congruence enumeration. The right Cayley graph $\Gamma_1$ is already complete, deterministic, and compatible with $R^\# \; \text{by definition.}$ It follows that $\text{TC}2$ and $\text{TC}3$ are the only steps actually applied in a congruence enumeration with input $\Gamma_1$. More precisely, $\text{TC}2(c)$ is applied to 0 and every $(u, v) \in S$, and then $\text{TC}3$ is repeatedly applied until $\kappa_i = \Delta_{N_i}$. The right Cayley graph of a monoid $M = \langle A \rangle$ of one of the types mentioned above, can also be computed using the Froidure-Pin Algorithm, which has complexity $O(|A||M|)$. Given that the right Cayley graph has $|M|$ nodes and $|A||M|$ edges, it is unlikely that, in general, there is a better method for finding the Cayley graph of a monoid from a generating set.

Experiments indicate that if the congruence being enumerated has a relatively large number of classes in comparison to $|M|$, then this second approach is faster than starting from the trivial word graph. On the other hand, if the number of congruence classes is relatively small compared with $|M|$, then starting from the trivial word graph is often faster. Since the number of congruence classes is usually not known in advance, the implementation in \textsc{libsemigroups} [24] runs both these approaches in parallel, accepting whichever provides an answer first.

We end this section with an example of performing a congruence enumeration with a right Cayley graph as an input.

**Example 5.2.** Suppose that $M$ is the monoid from Example 5.1. To compute the least right congruence $\rho$ on $M$ containing $S = \{(a, b)\}$ we perform the following steps.

Set $\Gamma_1$ to be the right Cayley graph of $M$ with respect to the generating set $\{a, b, c\}$; see Fig. 5.1 and Table 5.1. Suppose that $a < b < c$.

**Step 1:** Apply $\text{TC}2$ to the only pair $(a, b)$ in $S$, and the output of this is $(\Gamma_2, \kappa_2)$ where $\kappa_2$ is the least equivalence on the nodes of $\Gamma_2$ containing $(1, 2)$; see Fig. 5.2a.

**Step 2:** Apply $\text{TC}3$ to obtain the quotient of $\Gamma_2$ by $\kappa_2$, the output is $(\Gamma_3, \kappa_3)$ where $\Gamma_3 = \Gamma_2/\kappa_2$ and $\kappa_3$ is the least equivalence containing $(1, 5), (4, 6)$, and $(4, 7)$; see Fig. 5.2b.

**Step 3:** Apply $\text{TC}3$ with input $(\Gamma_3, \kappa_3)$ and output $(\Gamma_4, \kappa_4)$ where $\Gamma_4 = \Gamma_3/\kappa_3$ and $\kappa_4$ is the least equivalence containing $(4, 8)$; see Fig. 5.2c.

**Step 4:** Apply $\text{TC}3$ with input $(\Gamma_4, \kappa_4)$ and output $(\Gamma_5, \kappa_5)$ where $\Gamma_5 = \Gamma_4/\kappa_4$ and $\kappa_5 = \Delta_{N_5}$. Since $\Gamma_5$ is complete and compatible with $\rho$ and $\kappa_5$ is trivial, the enumeration terminates with $\Gamma_5$; see Fig. 5.2d.

We conclude that $\rho$ has 4 equivalence classes whose representatives $\varepsilon, a, c$, and $a^2$ correspond to the nodes $\{0, 1, 3, 4\}$ in the graph $\Gamma_5$. By following the paths in $\Gamma_5$ starting at 0 labelled by each of the words in Table 5.1 we determine that the congruence classes of $\rho$ are: $\{\varepsilon\}, \{(a, b, ab)\}, \{c\}, \{(a^2, ba, bc, bab)\}$. 


Figure 5.2: The output \((\Gamma_i, \kappa_i)\) of each step in Example 5.2. Purple arrows correspond to a, gray to b, pink to c, shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes; see Table 5.1 for a representative word for each node. A dashed edge with a single arrowhead denotes an edge that is obtained from TC2 or TC3, solid edges correspond to edges that existed at the previous step.
6 The HLT strategy

In this section, we describe the so-called HLT strategy for congruence enumeration. The acronym HLT stands for Haselgrove, Leech, and Trotter; for further details see [5] or [31, Section 5.2]. For the record, this is Walker’s Strategy 1 in [36], and is referred to as R-style in ACE [14] (“R” for “relators”).

The HLT strategy, like every congruence enumeration, starts by applying steps (a) and (b) in the definition (Definition 3.3). We then repeatedly apply TC3 until the resulting κ is trivial, after which we repeatedly perform the following steps:

HLT1. If α is the minimum node in Γ1 such that there exists (u, v) ∈ R where either: u or v does not label a path starting at α, or u and v label paths ending in different nodes. In the former case, we apply TC1 repeatedly until u and v label paths starting at α where v = vb for some b ∈ A, and then apply TC2 where part (a) applies. In the latter case, we simply apply TC2 to α and (u, v) ∈ R where part (c) applies.

HLT2. Repeatedly apply TC3 until the resulting κi+1 is trivial.

HLT3. If α is the node from HLT1, then we apply TC1 to α and every a ∈ A, if any, such that there is no edge incident to α labelled by a.

We note that if for every a ∈ A there exists (u, v) ∈ R such that either u or v starts with a, then HLT3 does not have to be performed.

Next we will show that the HLT strategy fulfils the definition of a congruence enumeration.

Proposition 6.1. If ⟨A|R⟩ is a finite monoid presentation, S is a finite subset of A∗ × A∗, R# is the least two-sided congruence on A∗ containing R, and ρ is the least right congruence on A∗ containing S and R#, then the HLT strategy applied to (Γ1, κ1) where Γ1 = (N1, E1) is a word graph such that (0)πΓ1 ⊆ ρ and κ1 = ΔN1 is a congruence enumeration.

Proof. It suffices to check that the conditions in Definition 3.3 hold. Clearly parts (a) and (b) are included in the HLT strategy and so there is nothing to check for these.

(c). Suppose that α is a node in Γi = (Ni, Ei) such that there is no edge labelled by some a ∈ A incident to α. If α ∈ Ni for all j ≥ i, then there exists k ∈ N such that α is the minimum node to which HLT1 is applied. If there exists (u, v) ∈ R such that the first letter of u is a, then an edge labelled by a incident to α is defined in HLT1. Otherwise, such an edge is defined in HLT3.

(d). Suppose that α is a node in Γi and (u, v) ∈ R are such that u and v both label paths starting at α. If α ∈ Ni for all j ≥ i, then there exists k ∈ N such that α is the minimum node to which HLT1 is applied, and so TC2 is applied to α and (u, v) at some later step. Hence either the paths labelled by u and v starting at α end at the same node, or they end at distinct nodes β and γ such that (β, γ) ∈ κi. But TC3 is applied at some later step, and so β and γ are identified in the corresponding quotient. Eventually the paths labelled by u and v starting at α end at the same node, i.e. (u, v) ∈ (α)πΓ1, for sufficiently large i.

(e). If (β, γ) ∈ κi for some i, then there exists j ≥ i such that TC3 is applied by the definition of HLT2. If we choose the smallest such j, then κi ⊆ κj and so (β, γ) /∈ κj+1.

Example 6.2. Let M be the monoid defined in Example 5.1. Then M is isomorphic to the monoid defined by the presentation

\[(a, b, c | ac = a^2, \ b^2 = b, \ ca = a^2, \ cb = bc, \ c^2 = a^2, \ a^3 = a^2, \ aba = a^2).\]

We will apply the HLT strategy to the presentation and the set S = \{(a, b)\}. In other words, we enumerate the least right congruence on the monoid M containing (a, b). This is the same right congruence that we enumerated in Example 5.1. The input word graph Γ1 is the trivial graph and the input κ1 is ΔN1. We suppose that a < b < c.

Step 1: We apply Definition 3.3(a). We add the nodes 1, 2 ∈ N1 and define the edges (0, a, 1) and (0, b, 2). Since S = \{(a, b)\} this concludes Definition 3.3(a). We continue by applying TC2 to 0 and (a, b). Since a and b label paths from 0 to the nodes 1 and 2, respectively, we define κ2 to be the least equivalence containing κ1 = ΔN1 and (1, 2). The output is graph Γ2 in Fig. 6.1a.

Step 2: We continue with Definition 3.3(b). We apply TC3 and we get the graph Γ3 in Fig. 6.1b which is the quotient of Γ2 by κ2. The set κ3 is defined to be ΔN3.
Now we are ready to apply Step 3: see Fig. 6.1. Table 6.1: A word labelling a path from 0 to each node in the right Cayley graph on the monoid $M$ by the presentation $\langle a, b, c | ac = a^2, b^2 = b, ca = a^2, cb = bc, c^2 = a^2, a^3 = a^2, aba = a^2 \rangle$.

|   | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $\varepsilon$ | a | b | $a^2$ | c |

Step 3: Now we are ready to apply HLT1. Since 0 is the minimum node in $\Gamma_3$ such that $(ac, a^2) \in R$ and $ac$ and $a^2$ do not label paths in $\Gamma_3$, we apply TC1 until node 3 and a path labelled by $a^2$ are defined and we continue by applying TC2(b) for $u = a^2, v = ac$ and $v_1 = a$. The output of this step is the graph $\Gamma_4$ in Fig. 6.1c.

Step 4-8: We apply HLT1 to the node 0 and each of the relations $(b^2, b), (ca, a^2), (cb, bc), (c^2, a^2), (a^3, a^2) \in R$ (in this order) and the output $(\Gamma_i, \kappa_i), i = 5, \ldots, 9$ are shown in Fig. 6.1d, Fig. 6.1e, Fig. 6.1f, Fig. 6.1g, and Fig. 6.1h.

Step 9: In this step, we do not apply HLT1 to 0 and $(aba, a^2)$ since $aba$ and $a^2$ both label paths that reach the same node in $\Gamma_9$. We apply HLT1 to the node 1 and the relation $(ac, a^2) \in R$ and the output is graph $\Gamma_{10}$ in Fig. 6.1i.

Step 10: We apply HLT1 to the node 1 and the relation $(cb, bc) \in R$ and the output is graph $\Gamma_{11}$ in Fig. 6.1j.

After Step 10, $\Gamma_{11}$ is complete, deterministic, and compatible with $R$. Hence the enumeration terminates; see Example 5.1 for details of the enumerated congruence.

7 The Felsch strategy

In this section, we describe two versions of the so-called Felsch strategy for congruence enumeration. For the record, this is Walker’s Strategy 2 in [36], and is referred to as C-style in ACE [14] (“C” for “cosets”).

7.1 First version

The Felsch strategy starts with the trivial word graph $\Gamma_1 = (N_1, E_1)$ and with the trivial equivalence $\kappa_1 = \Delta_{N_1}$ on the nodes $N_1$ of $\Gamma_1$. Just like the HLT strategy, the Felsch strategy starts by applying steps (a) and (b) from the definition of a congruence enumeration (Definition 3.3). We then repeatedly apply TC3 until the resulting $\kappa$ is trivial. The following steps are then repeatedly applied:

F1. If $\alpha \in N_i$ is the minimum node in $\Gamma_i$ such that there exists $a \in A$ and there is no edge with source $\alpha$ labelled by $a$, then apply TC1 to $\alpha$ and $a$. Apply TC2 to every node in $\Gamma_i$ and every relation in $R$.

F2. Apply TC3 repeatedly until $\kappa_{i+1}$ is trivial.

It follows immediately from the definition of the Felsch strategy that it satisfies the definition of a congruence enumeration. To illustrate the Felsch strategy, we repeat the calculation from Example 6.2.

Example 7.1. Let $M$ be the monoid defined by defined in Example 5.1. Then $M$ is isomorphic to the monoid defined by the presentation $(a, b, c | ac = a^2, b^2 = b, ca = a^2, cb = bc, c^2 = a^2, a^3 = a^2, aba = a^2)$.

We will apply the Felsch strategy to the presentation (with $a < b < c$) and the set $S = \{(a, b)\}$. In other words, we enumerate the least right congruence on the monoid $M$ containing $(a, b)$. This is the same right congruence that we enumerated in Example 5.1. The input word graph $\Gamma_1$ is the trivial graph and the input $\kappa_1$ is $\Delta_{N_1}$.

Step 1: We apply Definition 3.3(a). We add the nodes $1, 2 \in N_1$ and define the edges $(0, a, 1)$ and $(0, b, 2)$. Since $S = \{(a, b)\}$ this concludes Definition 3.3(a). We continue by applying TC2 to 0 and $(a, b)$. Since $a$ and $b$ label paths from 0 to the nodes 1 and 2, respectively, we define $\kappa_2$ to be the least equivalence containing $\kappa_1 = \Delta_{N_1}$ and $(1, 2)$. The output is graph $\Gamma_2$ in Fig. 7.1a.

Step 2: We continue with Definition 3.3(b). We apply TC3 and we get graph $\Gamma_3$ in Fig. 7.1b which is the quotient of $\Gamma_2$ by $\kappa_2$. The set $\kappa_3$ is defined to be $\Delta_{N_3}$.

Step 3: We apply F1 and hence we apply TC1 to 0 and $c$. We add the node $3 \in N_4$ and define the edge $(0, c, 3)$. The output is graph $\Gamma_4$ in Fig. 7.1c.
Figure 6.1: The output \((\Gamma_i, \kappa_i)\) of each step in the HLT strategy in Example 6.2. Purple arrows correspond to \(a\), gray to \(b\), pink to \(c\), shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes; see Table 6.1 for a representative word for each node. A dashed edge with a double arrowhead indicates the edge being defined in TC1, a dashed edge with a single arrowhead denotes an edge that is obtained from TC2 or TC3, solid edges correspond to edges that existed at the previous step.
Figure 7.1: The output \((\Gamma_i, \kappa_i)\) of each step in the Felsch strategy in Example 7.1. Purple arrows correspond to \(a\), gray to \(b\), pink to \(c\), shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes; see Table 7.1 for a representative word for each node. A dashed edge with a double arrowhead indicates the edge being defined in TC1, a dashed edge with a single arrowhead denotes an edge that is obtained from TC2 or TC3, solid edges correspond to edges that existed at the previous step.

Table 7.1: A word labelling a path from 0 to each node in the right Cayley graph on the monoid \(M\) from Example 7.1; see Fig. 7.1

Step 4: We continue with the application of the second part of \(F1\) and hence we apply TC2 to 1 and \((b^2, b)\). The output is graph \(\Gamma_5\) in Fig. 7.1d.

Step 5: We apply \(F1\) and hence we apply TC1 to 1 and \(a\). We add the node 4 \(\in N_6\) and define the edge \((1, a, 4)\). The output is graph \(\Gamma_6\) in Fig. 7.1e.

Steps 6-12: We continue with the application of the second part of \(F1\) and hence we apply TC2 to 0 and \((a^3, a^2)\), to 0 and \((ac, a^2)\), to 0 and \((ac, a^2)\), to 0 and \((c^a, a^2)\) and to 3 and \((b^2, b)\).

After Step 12, \(\Gamma_{13}\) in Fig. 7.1f is complete, deterministic, and compatible with \(R\). Hence the enumeration terminates; see Example 5.1 for details of the enumerated congruence.

7.2 Second version

The purpose of \(F2\) in the Felsch strategy is to squeeze as much information as possible out of every definition of an edge \((\alpha, a, \beta)\) made in \(F1\). The implementation of the Felsch strategy in [23] spends most of its time performing TC2. In this section, we propose a means of reducing the number of times TC2 is applied in \(F2\) of the Felsch strategy. Roughly speaking, we do this by only applying TC2 to \((u, v) \in R\) and a node \(\alpha\) in \(\Gamma_i\) if the path in \(\Gamma_i\) starting at \(\alpha\) and labelled by \(u\), or \(v\), goes through a part of \(\Gamma_i\) that has recently changed.

To enable us to do this, we require a new set \(D_i \subseteq N_i \times A\), in addition to the word graph \(\Gamma_i\) and equivalence relation \(\kappa_i\), at every step of a congruence enumeration. An element \((\alpha, a)\) of \(D_i\) corresponds to a recently defined edge incident to \(\alpha\) and labelled by \(a\). More precisely, we define \(D_1 = \emptyset\) and for \(i \geq 1\) we define

\[
D_{i+1} := \{(\alpha, a) \in N_i \times A : (\alpha, a, \beta) \in E_{i+1} \setminus E_i\}.
\]

In order to efficiently use the information recorded in \(D_i\), we require the following definition.

Definition 7.2. If \((A|R)\) is a finite monoid presentation, then we define the Felsch tree \(F(A, R)\) of this presentation to be a pair \((\Theta, \iota)\) where:
Figure 7.2: The Felsch tree $F(A, R)$ from Example 7.3.

(a) $\Theta = (N, E)$ is a graph with nodes $N$ consisting of every (contiguous) subword $w \in A^*$ of a word in a relation in $R$ and edges $(u, a, au) \in E$ whenever $u, au \in N$ and $a \in A$; and

(b) a function $\iota$ from $N$ to the power set $\mathcal{P}(R)$ of $R$ such that $(u, v) \in (w)\iota$ whenever $w \in N$ is a prefix of $u$ or $v$.

**Example 7.3.** If $A = \{a, b\}$ and $R = \{(a^4, a), (b^3, b), ((ab)^2, a^2)\}$, then the nodes in the word graph $\Theta$ in $F(A, R)$ are

$$N = \{\varepsilon, a, b, a^2, ab, ba, b^2, a^3, aba, bab, b^3, a^4, (ab)^2\}.$$  

A diagram of $\Theta$ can be found in Fig. 7.2.

The function $\iota : N \to \mathcal{P}(R)$ is:

\[
\begin{align*}
(\varepsilon)\iota &= R, \\
(a)\iota = (a^2)\iota &= \{(a^4, a), ((ab)^2, a^2)\} \\
(a^3)\iota = (a^4)\iota &= \{(a^4, a)\} \\
(aba)\iota &= \{((ab)^2, a^2)\} \\
(b)\iota = (b^2)\iota = (b^3)\iota &= \{\{b^3, b\}\} \\
(ab)\iota = ((ab)^2)\iota &= \{((ab)^2, a^2)\} \\
(ba)\iota = (bab)\iota &= \emptyset.
\end{align*}
\]

We modify $\text{TC1}$, $\text{TC2}$, and $\text{TC3}$ so that the set $D_i$ is defined appropriately at every step, and then, roughly speaking, we replace $F_2$ in the Felsch strategy by a backtrack search through $\Theta$ in $F(A, R)$ for every pair in $D_i$. We will refer to this as the modified Felsch strategy.

If $\alpha$ is a node in $\Gamma_i$ and $v \in A^*$ is a node in $\Theta$, then we perform a backtrack search consisting of the following steps:

**PD1.** Apply $\text{TC2}$ to $\alpha$ and every relation in $(v)\iota$.

**PD2.** For every edge $(v, b, bv)$ in $\Theta$ that has not been traversed: for every node $\beta$ in $\Gamma_i$ such that $(\beta, b, \alpha) \in E_i$, apply **PD1** and **PD2** to $\beta$ and $bv$.

**PD3.** Repeatedly apply $\text{TC3}$ until $\kappa$ is trivial.

We will refer to either removing a pair from $D_i$ or performing the backtrack search as processing a deduction.

The backtrack search is initiated for every $(\alpha, a) \in D_i$. If $a$ is not a node in $\Theta$, then $a$ does not occur in any relation in $R$ and so following the path in $\Gamma_i$ labelled by any $u$ such that $(u, v) \in R$ from any node in $w$ in $\Gamma_i$ cannot contain any
edge labelled by $a$. In particular, no such path contains the edge $(\alpha, a, \beta) \in \Gamma$; the definition of which caused $(\alpha, a)$ to belong to $D_i$ in the first place. Hence if $(\alpha, a) \in D_i$, then there is a node $a$ in $\Theta$.

We repeatedly apply $PD_1$, $PD_2$, and $PD_3$ starting from every pair in $D_i$. This must terminate eventually because $\Theta$ is finite and we only apply $TC_2$ and $TC_3$ in $PD_1$, $PD_2$, and $PD_3$.

To show that the modified Felsch strategy is a congruence enumeration, we require the following lemma.

**Lemma 7.4.** Suppose that $\alpha$ is a node in $\Gamma_j$ for all $j \geq i$, and that $(u, v) \in R$ is such that $u$ and $v$ label paths $P_u$ and $P_v$ in $\Gamma_i$ starting at $\alpha$. If $(\beta, a) \in D_i$ for some $i$, and there is an edge $(\beta, a, \gamma) \in \Gamma_i$ in either $P_u$ or $P_v$, then $(u, v) \in (\alpha)_{\pi_{\Gamma_k}}$ for some $k \geq i$.

**Proof.** Suppose that $P_u$ consists of the edges $(\mu_1, b_1, \mu_2, \ldots), (\mu_r, b_r, \mu_{r+1})$ where $\mu_1 = \alpha$ and $u = b_1 \cdots b_r$. Similarly, suppose that $P_v$ consists of the edges $(\nu_1, c_1, \nu_2, \ldots), (\nu_s, c_s, \nu_{s+1})$ where $\nu_1 = \alpha$ and $v = c_1 \cdots c_s$. We may assume that $\mu_{r+1} \neq \nu_{s+1}$ and that there exists $t \in \{1, \ldots, r\}$ such that $(\mu_t, b_t, \mu_{t+1}) = (\beta, a, \gamma)$.

Since $(\beta, a) \in D_i$, the backtrack search in $PD_1$ and $PD_2$ will be applied to $(\beta, a)$ at some step $j \geq i$ of the enumeration. This search begins with $PD_1$, where $TC_2$ is applied to $\beta$ and every relation in $(\alpha)_{\pi_{\Gamma_k}}$. At step $PD_2$ we apply $PD_1$ and $PD_2$ to every $\zeta \in N_i$ and $w$ such that there exists an edge of the form $(a, b, w) \in \Theta$ and an edge $(\zeta, b, \beta) \in E_i$.

By definition, $u = b_1 \cdots b_{i-1}a b_{i+1} \cdots b_r$. If $m \in \{1, \ldots, t\}$, then, since $(u, v) \in R$, $b_m \cdots b_{i-1}a$ and $b_m \cdots b_{i-1}a$ are nodes of $\Theta$ and there exists an edge $(b_m \cdots b_{i-1}a, b_{i-1}a, b_m \cdots b_{i-1}a)$ in $\Theta$. Also by assumption $(\mu_m - 1, b_m - 1, \mu_m) \in E_i$ and $(\mu_m - 1, b_m - 1, \mu_m)$ is one of the pairs in $PD_1$ if it is applied to $(\mu_m, b_m - 1, \mu_m - 1)$. If $m = t$, then $(\mu_m - 1, b_m - 1, \mu_m) = (\mu_t, a) = (\beta, a)$ and hence $PD_2$ is applied to $(\mu_m - 1, b_m - 1, \mu_m - 1)$ for every $m \in \{1, \ldots, t\}$. In particular, $PD_1$ is applied to $(\mu_1, b_1 \cdots b_{i-1}a)$ at some point. This means that $TC_2$ is applied to $\alpha$ and every relation in $(\beta, a, \beta)$.

Since $(\beta, a, \beta)$ consists of the edges $(\alpha, \beta, a, \gamma)$, it suffices to prove that $(u, v) \in (\alpha)_{\pi_{\Gamma_k}}$ for some $k \geq i$. Because $u$ and $v$ were assumed to label paths originating at $\alpha$, neither $TC_2(a)$ nor $(b)$ applies. Additionally, we assumed that $\mu_{r+1} \neq \nu_{s+1}$ and so $TC_2(c)$ applies and $(\mu_{r+1}, \nu_{s+1}) \in \kappa_{i+1}$.

As noted above, $\Theta$ is finite, and $PD_1$ and $PD_2$ are only applied finitely many times before there is an application of $PD_3$. If the first application of $TC_3$ (in $PD_3$) after step $j$ occurs at step $k$, then $\Gamma_{k+1}$ is the quotient of $\Gamma_k$ by $\kappa_k$ and $\kappa_i \subseteq \kappa_k$. Therefore $(u, v) \in (\alpha)_{\pi_{\Gamma_{k+1}}}$ as required.

**Proposition 7.5.** If $(A, R)$ is a finite monoid presentation, $S$ is a finite subset of $A^* \times A^*$, $R^\#$ is the least two-sided congruence on $A^*$ containing $R$, and $\rho$ is the least right congruence on $A^*$ containing $S$ and $R^\#$, then the modified Felsch strategy applied to $((\Gamma_1, \kappa_1))$, where $\Gamma_1 = (N_1, E_1)$ is the trivial word graph and $\kappa_1 = \Delta_{N_1}$, is a congruence enumeration.

**Proof.** The only difference between the modified Felsch strategy and the original Felsch strategy is that after $F_1$, $TC_2$ is only applied to particular nodes and relations. Hence, it suffices to show that Definition 3.3(d) holds. Assume $\alpha \in N_i$ at some step $i$ in the congruence enumeration and let $(u, v) \in R$. In order for modified Felsch to be a congruence enumeration we need to show that there exists $j \geq i$ such that either $\alpha \notin N_j$ or $(u, v) \in (\alpha)_{\pi_{\Gamma_j}}$. If $\alpha \notin N_j$ for some $j > i$, then $\alpha \notin N_k$ for all $k \geq j$ (since new nodes introduced in $TC_1$ are larger than all previous nodes). Hence it suffices to prove that if $\alpha$ is a node for all $j \geq i$, there exists some $k$ such that $(u, v) \in (\alpha)_{\pi_{\Gamma_k}}$. Since $F_1$ is repeatedly applied there exists a step $k + 1$ when $\alpha$ is a node in $\Gamma_{k+1}$ and paths $P_u$ and $P_v$ leaving $\alpha$ labelled by $u$ and $v$, respectively, exist in $\Gamma_{k+1}$. Suppose that $k \in \mathbb{N}$ is the least value such that this holds. Then, by Lemma 7.4, it suffices to show that there exists an edge $(\beta, a, \gamma)$ in either $P_u$ or $P_v$ that belongs to $E_{k+1} \setminus E_k$ so that $(\beta, a) \in D_{k+1}$. If every edge in $P_u$ and $P_v$ belongs to $E_k$, then $u$ and $v$ both label paths in $\Gamma_k$ starting at $\alpha$, which contradicts the assumed minimality of $k$. 

8 Implementation issues

In this section, we briefly address some issues relating to any implementation of the Todd-Coxeter algorithm for semigroups and monoids as described herein.

In some examples, it can be observed that the HLT strategy defines many more nodes than the Felsch strategy. One possible antidote to this is to, roughly speaking, perform

“periods of définition à la HLT [that] alternate with periods of intensive scan à la Felsch” [25, p. 14].

In both ACE [14] and libsemigroups [24] it is possible to specify the precise lengths of the periods of applications of HLT and Felsch. As might be expected, some choices work better than others in particular examples, and this is difficult (or impossible) to predict in advance. It is routine to show that alternating between HLT and Felsch in this way still meets the definition of a congruence enumeration given in Definition 3.3. Although we presented the HLT and Felsch strategies separately, it seems that some combination of the two sometimes offers better performance.
The next issue is: how to represent the equivalence relations \( \kappa_i \)? A method suggested in [31, Section 4.6], which is now a standard approach for representing equivalence relations, is to use the disjoint-sets data structure to represent the least equivalence relation containing the pairs \((u, v)\) added to \( \kappa_i \) in TC2(c) or TC3. The theoretical time complexity of updating the data structure to merge two classes, or to find a canonical representative of a class given another representative, is \( O(\alpha(m)) \) time (in both the worst and the average case) and requires \( O(m) \) space where \( m \) is the number of elements in the underlying set and \( \alpha \) is the inverse Ackermann function; see [34].

Another issue that arises in the implementation is how to represent the word graphs \( \Gamma_i \). In order to efficiently obtain the word graph \( \Gamma_{i+1} \) from \( \Gamma_i \) in TC3 it is necessary to keep track of both the edges with given source node, and with given target. This is more complex for semigroups and monoids than for groups, because a word graph \( \Gamma = (N, E) \) output by a coset enumeration for a group has the property that for every \( \beta \in N \) and every \( a \in A \) there is exactly one \( \alpha \in N \) such that \( (\alpha, a, \beta) \) is an edge. As such if it ever arises that there are edges \((\alpha_1, a, \beta)\) and \((\alpha_2, a, \beta)\) in a word graph during a coset enumeration, the pair \((\alpha_1, \alpha_2)\) can immediately be added to \( \kappa_i \). It is therefore possible to represent every word graph arising during a coset enumeration to have the property that \(|\{ \alpha \in N : (\alpha, a, \beta) \in E \}| = 1\), which simplifies the data structure required to represent such a graph.

In contrast, if \( \Gamma = (N, E) \) is a word graph arising in a congruence enumeration for a monoid, then \(|\{ \alpha \in N : (\alpha, a, \beta) \in E \}| \) can be as large as \(|N|\). In practice, in TC3 pairs of nodes belonging to \( \kappa_i \) are merged successively. A balance must be struck between repeatedly updating the data structure for the edges with given target in \( \Gamma_{i+1} \) or only retaining the edges with given source and rebuilding the data structure for the target edges later in the process. The former works better if the number of nodes in \( \Gamma_i \) is comparable to the number in \( \Gamma_{i+1} \), i.e. only a relatively small number of nodes are merged. On the other hand, if \( \Gamma_{i+1} \) is considerably smaller than \( \Gamma_i \), then it can be significantly faster to do the latter.

Depending on the sequence of applications of TC1, TC2, and TC3 in two successful congruence enumerations with the same input, the output word graphs may not be equal. However, the output word graphs are unique up to isomorphism. **Standardization** is a process for transforming a word graph into a standard form. To discuss this we require the following notion.

If \( A \) is any alphabet and \( \preceq \) is a total order on \( A^* \), then we say that \( \preceq \) is a **reduction ordering** if \( \preceq \) has no infinite descending chains and if \( u \preceq v \) for some \( u, v \in A^* \), then \( puq \preceq pvq \) for all \( p, q \in A^* \). It follows from this definition that \( \varepsilon \) is the \( \preceq \)-minimum word in \( A^* \) for every reduction ordering on \( A^* \).

If the set \( A \) is totally ordered by \( \preceq \), then we may extend \( \preceq \) to \( \preceq_{\text{lex}} \) over \( A^* \), by defining \( \varepsilon \preceq_{\text{lex}} w \) for all \( w \in A^* \) and \( u \preceq_{\text{lex}} v \) whenever \( u = au_1 \) and \( v = bv_1 \) for some \( a, b \in A \) with \( a < b \), or \( a = b \) and \( u_1 \preceq_{\text{lex}} v_1 \). This order is usually referred to as the **lexicographic order** on \( A^* \). Note that lexicographic order is not a reduction ordering. The **short-lex order** \( \preceq_{\text{lex}} \) on \( A^* \) is defined as follows: if \( u, v \in A^* \), then \( u \preceq_{\text{lex}} v \) if \( |u| < |v| \) or \( |u| = |v| \) and \( u \preceq \) \( v \). It is straightforward to verify that the short-lex order on \( A^* \) is a reduction ordering. Further examples of reduction orderings on \( A^* \) include recursive path descent, as well as the wreath product of any finite collection of reduction orderings; see [31, Section 2.1] for further details.

Suppose that \( \preceq \) is a reduction ordering on \( A^* \). We will say that the word graph \( \Gamma = (N, E) \) is **standardized with respect to** \( \preceq \) if \( \alpha < \beta \) if and only if \( w_\alpha < w_\beta \) for any \( \alpha, \beta \in N \) where \( w_\alpha, w_\beta \in A^* \) are the \( \preceq \)-minimum words labelling \((0, \alpha)\)- and \((0, \beta)\)-paths, respectively. Any process that transforms a word graph \( \Gamma \) into a standardized word graph, is referred to as **standardization**; for example, see STANDARDIZE in [31, p195].

A word graph \( \Gamma_i \) can be replaced by any standardized word graph at any step of a congruence enumeration, provided that the values in \( \kappa_i \) and \( D_i \) are also updated accordingly. In particular, it can be applied repeatedly during a congruence enumeration, or only at the end. Standardization during a congruence enumeration can be very costly in the context of semigroups and monoids. However, it can also be somewhat beneficial in some examples.

The order of the definitions of new nodes in \( \Gamma_{i+1} \) in both HLT1 and F1 depends on the numerical values of their source nodes. Hence replacing \( \Gamma_i \) by a standardized word graph can change the order of these definitions, which in turn can influence the number of steps in the enumeration. In an actual implementation, standardising a word graph is a rather complicated process that involves applying a permutation to the data structure representing the graph.

### 9 Further variants

In this section, we present some variants of the Todd-Coxeter algorithm that appear in the literature, which are used to compute special types of congruences (namely Rees congruences) and for computing congruences on finitely presented inverse monoids.
9.1 Monoids with zero

The first such variant is for finitely presented monoids with a zero element 0. For example, if \( M \) is the monoid defined by the presentation \( \langle a, b \mid ab = 0, a^4 = a, b^3 = b, (ab)^2 = 0, a0 = 0a = 0 = b0 = 0b = 0^2 \rangle \),

then the relations \( a0 = 0a = 0 = b0 = 0b = 0^2 \) indicate that 0 is a zero element of \( M \). Of course, the algorithms described above can be applied to this finite monoid presentation, as well as every other. On the other hand, the inclusion of the relations \( a0 = 0a = 0 = b0 = 0b = 0^2 \) is rather cumbersome, and so we might rather write:

\[ \langle a, b \mid ab = 0, a^4 = a, b^3 = b, (ab)^2 = 0 \rangle \]

where the relations \( a0 = 0a = 0 = b0 = 0b = 0^2 \) are implicit. This is directly analogous to the implicit relations for the identity in a monoid presentation, and in a group presentation for inverses. We refer to such a presentation as a **finite monoid-with-0 presentation**. Both the HLT and Felsch strategies can be adapted for monoid-with-0 presentations without much difficulty as follows.

Suppose that \( \langle A|R \rangle \) is a finite monoid-with-0 presentation. We refer to a word graph \( \Gamma_i = (N_i, E_i) \) over \( A \cup \{0\} \) with a distinguished node \( \omega \in N_i \) such that the only edges with source \( \omega \) are loops of the form \( (\omega, a, \omega) \in E_i \) for all \( a \in A \) and edges \( (\alpha, 0, \omega) \) for all \( \alpha \in N_i \) as a **word graph-with-0**. We augment **TC1** with the following step:

**Z:** If \( \beta \) is the new node introduced in **TC1**, then we define the edge \( (\beta, 0, \omega) \).

Given a monoid-with-0 presentation, it is routine to verify that if we perform any congruence enumeration (where **TC1** is augmented with **Z**) with input \( (\Gamma_1, \kappa_1) \) where \( \Gamma_1 = (N_1, E_1) \) is a word graph-with-0 and \( \kappa_1 = \Delta_{N_1} \), then the conclusions in Corollary 4.2 still hold.

9.2 Rees congruences

Following [30, Chapter 12], we may extend the discussion of the previous section, to obtain a procedure for enumerating a left, right, or two-sided Rees congruence on a finitely presented monoid (with or without zero element). If \( I \) is a left, right, or two-sided ideal of the monoid \( S \), then the **Rees congruence** associated with \( I \) is the congruence \( \Delta_S \cup (I \times I) \). Such a procedure applies to a finite monoid presentation \( \langle A|R \rangle \) and set finite \( S \subseteq A^* \times \{0\} \) (rather than \( S \subseteq A^* \times A^* \) as in Definition 3.3). The input to such an enumeration is a word graph-with-zero \( \Gamma_1 = (N_1, E_1) \) and \( \kappa_1 = \Delta_{N_1} \). The first steps are identical to those given in Definition 3.3(a) and (b) except that **TC1** and **Z** are applied in part (a). The subsequent steps are just any sequence of applications of **TC1+Z**, **TC2**, and **TC3** satisfying Definition 3.3(c), (d) and (e). It follows immediately from Theorem 4.1, and the validity of the congruence enumeration for monoid-with-0 presentations, that this process is valid.

9.3 Stephen’s procedure

Another variant of the Todd-Coxeter algorithm is that of Stephen [32, Chapter 4], mentioned in the introduction of the current article. Note that a similar method for constructing the Cayley graph of groups was described by Dehn [6]. Suppose that \( M \) is the monoid defined by a finite monoid presentation \( \langle A|R \rangle \) and that \( \Gamma = (N, E) \) is the right Cayley graph of \( M \) with respect to \( A \). If \( w \in A^* \) is arbitrary and \( w \) labels a \((0, \alpha)\)-path in \( \Gamma \) for some \( \alpha \in N \), then the aim of this variant is to output the subgraph \( \Lambda \) of \( \Gamma \) induced by the set \( X \) of nodes in \( N \) from which \( \alpha \) is reachable. Note that the set of these nodes corresponds to the set of elements in \( M \) which are \( \geq_R w/R^\# \) (recall that two monoid elements \( s, t \in M \) satisfy \( s \geq_R t \) if \( tM \subseteq sM \)). If \( A \) is the automata with alphabet \( A \), state set \( X \), initial state 0, accept state \( \alpha \), and edges consisting of those in \( \Lambda \), then the language \( L(A) \) accepted by \( A \) is the set of words in \( v \in A^* \) that represent the same element of \( M \) as \( w \) (i.e. \( v/R^\# = w/R^\# \)). As such if the (as yet to be described) procedure terminates, the output allows us to decide the word problem for \( w \) in \( M \).

Suppose that \( w = a_1 \cdots a_n \in A^* \) for some \( a_1, \ldots, a_n \in A \), that \( \Gamma_1 = (N_1, E_1) \) is the trivial word graph, and \( \kappa_1 = \Delta_{N_1} \). A special case of Stephen’s procedure consists of the following steps described in terms of **TC1**, **TC2**, and **TC3**. The first step is always:

**S1:** **TC1** is applied to 0 and \( a_1 \), then to \( i \) and \( a_{i+1} \) for every \( i \) such that \( 1 \leq i \leq n - 1 \). The resulting \( \Gamma_n \) consists of the single path from 0 to the node \( n \). (The graph \( \Gamma_n \) is referred to as the **linear graph of** \( w \) in [32].)

**S1** is then followed by any sequence of the following steps:
S2: At step $i$, suppose that the word graph $\Gamma_i = (N_i, E_i)$ contains a path with source $\alpha \in N_i$ labelled $u$ for some $(u, v) \in R$. If $v = v_1 b$ where $v_1 \in A^*$ and $b \in A$, then TC1 is applied until there is a path with source $\alpha$ labelled by $v_1$ and then TC2 is applied to $\alpha$ and $(u, v) \in R$. (This is referred to as an elementary expansion in [32].)

S3: Apply TC3. (Quotienting $\Gamma_i$ by the least equivalence containing a single $(\alpha, \beta) \in N_i \times N_i$ is referred to as a determination in [32]. It is shown in [32] that if any sequence of S2 and S3 has the property that after finitely many steps any subsequent applications of S2 and S3 result in no changes to the output (i.e. $(\Gamma_{i+1}, \kappa_{i+1}) = (\Gamma_i, \kappa_i)$), then $\Gamma_i$ is isomorphic to the induced subgraph $\Lambda$ defined at the start of this section. Note that S2 and S3 are similar to HLT1, HLT2, and HLT3 described in Section 6. While it is not possible to use congruence enumeration, at least as described in this paper, to solve the word problem when the monoid $M$ defined by a presentation $(A|R)$ is infinite, it is possible to decide whether or not two words $u, v \in A^*$ represent the same element of $M$ using the procedure defined in this section, whenever the induced subgraph $\Lambda$ is finite. Since the set of nodes in $\Lambda$ corresponds to the set of elements in $M$ which are $\geq_R u/R\#$ (assuming that $u$ is the input word for Stephen’s procedure) the word graph $\Lambda$ is finite when there are only finitely many elements of $M$ that are $\geq_R u/R\#$.

Acknowledgements

The authors would like to thank the referee for their careful reading of the paper, and for their helpful suggestions. The third author was supported by the EPSRC doctoral training partnership number EP/N509759/1 when working on this project. The fourth author would like to thank the School of Mathematics and Statistics of the University of St Andrews and the Cyprus State Scholarship Foundation for their financial support.

References

[1] A. Abram and C. Reutenauer. “The Stylic Monoid”. In: Semigroup Forum 105 (Aug. 2022), pp. 1–45. DOI: 10.1007/s00233-022-10285-3.

[2] R. E. Arthur and N. Ruškuc. “Presentations for Two Extensions of the Monoid of Order-Preserving Mappings on a Finite Chain”. In: Southeast Asian Bulletin of Mathematics 24.1 (Mar. 2000), pp. 1–7. DOI: 10.1007/a10012-000-0001-1.

[3] W. W. Boone. “The word problem”. In: Proceedings of the National Academy of Sciences 44.10 (Oct. 1958), pp. 1061–1065. DOI: 10.1073/pnas.44.10.1061.

[4] A. J. Cain and V. Maltcev. “Decision problems for finitely presented and one-relation semigroups and monoids”. In: International Journal of Algebra and Computation 19.06 (Sept. 2009), pp. 747–770. DOI: 10.1142/s0218196709005366. URL: https://doi.org/10.1142/s0218196709005366.

[5] J. J. Cannon, L. A. Dimino, G. Havas, and J. M. Watson. “Implementation and analysis of the Todd-Coxeter algorithm”. In: Math. Comp. 27 (1973), pp. 463–490. ISSN: 0025-5718. DOI: 10.2307/2005654.

[6] M. Dehn. “Über unendliche diskontinuierliche Gruppen”. In: Mathematische Annalen 71 (1911), pp. 116–144. DOI: 10.1007/BF01456932.

[7] D. Easdown, J. East, and D. G. Fitzgerald. “A presentation for the dual symmetric inverse monoid”. In: 18.02 (Mar. 2008), pp. 357–374. DOI: 10.1142/s0218196708004470.

[8] J. East. “Generators and relations for partition monoids and algebras”. In: Journal of Algebra 339.1 (Aug. 2011), pp. 1–26. DOI: 10.1016/j.jalgebra.2011.04.008.

[9] J. East. “Presentations for Temperley–Lieb Algebras”. In: The Quarterly Journal of Mathematics (Feb. 2021). DOI: 10.1093/qmath/haab001.

[10] D. G. FitzGerald. “A presentation for the monoid of uniform block permutations”. In: Bulletin of the Australian Mathematical Society 68.2 (2003), pp. 317–324. DOI: 10.1017/S0004972700037692.

[11] V. Froidure and J.-E. Pin. “Algorithms for computing finite semigroups”. In: Foundations of computational mathematics (Rio de Janeiro, 1997). Berlin: Springer, 1997, pp. 112–126.

[12] GAP – Groups, Algorithms, and Programming, Version 4.12.2. The GAP Group. 2022. URL: https://www.gap-system.org.
[13] J. Gay and F. Hivert. "The 0-Rook Monoid and its Representation Theory." 2019. arXiv: 1910.11740 [math.CO].

[14] G. Havas and C. Ramsay. "Coset enumeration: ACE version 5.2." http://staff.itee.uq.edu.au/havas. 2009.

[15] D. F. Holt, B. Eick, and E. A. O’Brien. Handbook of computational group theory. Boca Raton, Ann Arbor, London, Tokyo: CRC Press, 2004.

[16] J. M. Howie. Fundamentals of semigroup theory. Vol. 12. London Mathematical Society Monographs. New Series. Oxford Science Publications. New York: The Clarendon Press Oxford University Press, 1995, pp. x+351. ISBN: 0-19-851194-9.

[17] A. Jura. “Coset enumeration in a finitely presented semigroup”. In: Canad. Math. Bull. 21.1 (1978), pp. 37–46. ISSN: 0008-4395.

[18] J. Leech. “Coset enumeration on digital computers”. In: Mathematical Proceedings of the Cambridge Philosophical Society 59.02 (1963), p. 257. DOI: 10.1017/s0305004100036872.

[19] S. A. Linton. “Constructing matrix representations of finitely presented groups”. In: J. Symbolic Comput. 12.4-5 (1991). Computational group theory, Part 2, pp. 427–438. ISSN: 0747-7717. DOI: 10.1016/S0747-7717(08)80095-8.

[20] S. A. Linton. “On vector enumeration”. In: Linear Algebra Appl. 192 (1993). Computational linear algebra in algebraic and related problems (Essen, 1992), pp. 235–248. ISSN: 0024-3795. DOI: 10.1016/0024-3795(93)90245-J.

[21] V. Maltcev and V. Mazorchuk. “Presentation of the singular part of the Brauer monoid”. In: Mathematica Bohemica 132.3 (2007), pp. 297–323. DOI: 10.21136/mb.2007.134125.

[22] A. A. Markov. “On the impossibility of certain algorithms in the theory of associative systems”. In: Dokl. Akad. Sci. USSR 55 (1947), pp. 587–590.

[23] J. D. Mitchell et al. An implementation of the Todd-Coxeter Algorithm for Semigroups and Monoids in python3. https://gist.github.com/james-d-mitchell/6b06bc78e2bcbdb6def53a2654d9f953. 2021.

[24] J. D. Mitchell et al. libsemigroups - C++ library for semigroups and monoids, Version 2.7.3.

[25] J. Neubüser. “An elementary introduction to coset table methods in computational group theory”. In: Groups - St Andrews 1981. Ed. by C. M. Campbell and E. F. Robertson. London Mathematical Society Lecture Notes Series. Cambridge University Press, 1982, pp. 1–45. DOI: 10.1017/CBO9780511661884.004.

[26] B. H. Neumann. “Some remarks on semigroup presentations.” English. In: Can. J. Math. 19 (1968), pp. 1018–1026. ISSN: 0008-414X; 1496-4279/e.

[27] P. S. Novikov. “On the algorithmic unsolvability of the word problem in group theory”. Russian. In: Tr. Mat. Inst. Steklova 44.140 (1955).

[28] E. F. Robertson and Y. Ünlü. “On Semigroup Presentations”. In: Proceedings of the Edinburgh Mathematical Society 36 (1992), pp. 55–68.

[29] J. J. Rotman. An introduction to the theory of groups. Fourth. Vol. 148. Graduate Texts in Mathematics. Springer-Verlag, New York, 1995, pp. xvi+513. ISBN: 0-387-94285-8. URL: https://doi.org/10.1007/978-1-4612-4176-8.

[30] N. Ruškuc. “Semigroup presentations”. PhD thesis. University of St Andrews, 1995.

[31] C. C. Sims. Computation with finitely presented groups. Encyclopedia of mathematics and its applications. Cambridge, England, New York: Cambridge University Press, 1994. ISBN: 0-521-43213-8. URL: http://opac.inria.fr/record=b1082972.

[32] J. B. Stephen. “Applications of automata theory to presentations of monoids and inverse monoids”. PhD thesis. University of Nebraska, 1987. URL: https://digitalcommons.unl.edu/dissertations/AAI8803771.

[33] J. Stephen. “Presentations of inverse monoids”. In: Journal of Pure and Applied Algebra 63.1 (1990), pp. 81–112.

[34] R. E. Tarjan. “Efficiency of a Good But Not Linear Set Union Algorithm”. In: Journal of the ACM 22.2 (Apr. 1975), pp. 215–225. DOI: 10.1145/321879.321884.

[35] J. A. Todd and H. S. M. Coxeter. “A practical method for enumerating cosets of a finite abstract group”. In: Proceedings of the Edinburgh Mathematical Society 5.01 (1936), pp. 26–34. DOI: 10.1017/s0013091500008221. URL: https://doi.org/10.1017/s0013091500008221.

[36] T. Walker. “Semigroup enumeration - Computer implementation and Applications”. PhD thesis. University of St Andrews, 1992.

[37] T. Walker. tcsemi. 1992. URL: https://github.com/james-d-mitchell/tcsemi.

[38] A. Williams. Monoid automata factory. 2016. URL: http://maffsa.sourceforge.net/.
A  Extended examples

This appendix contains a number of extended examples of congruence enumerations.

Example A.1. We will apply the Felsch strategy to the presentation

\[ P = \langle a, b \mid a^3 = a, b^3 = b, (ab)^2 = a^2 \rangle \]

with \( a < b \). The input word graph \( \Gamma_1 \) is the trivial graph and the input \( \kappa_1 = \Delta_{N_1} \). Since \( S = \emptyset \), we do not apply steps Definition 3.3(a) and (b).

For the sake of simplicity, the steps in this example correspond to either a single application of \( \text{TC1} \) (a single application of \( \text{TC1} \)) and multiple applications of \( \text{TC2} \) (or a single application of \( \text{TC3} \). If a step produces no change to \( \Gamma_i \) or \( \kappa_i \), this step is skipped and does not have a number; see Table A.1, Fig. A.1, Fig. A.2, Fig. A.3, Fig. A.4, and Fig. A.5.

Step 1-2: The only node in \( \Gamma_1 \) is 0. Since there is no edge incident to 0 labelled by \( a \), in Step 1 we apply \( \text{F1} \) and add the node 1 and the edge \( (0, a, 1) \). Similarly, in Step 2 we add the node 2 and the edge \( (0, b, 2) \). The output is the word graph \( \Gamma_2 \) in Fig. A.1a.

Step 3: An application of \( \text{TC1} \) which leads to the definition of the node 3 and the edge \( (1, a, 3) \). At this point \( \text{F1} \) leads to an application of \( \text{TC2} \) to the node 3 and the relation \( (a^3, a) \) and hence we define the edge \( (3, a, 1) \). The output of step 3 is \( \Gamma_3 \); see Fig. A.1b.

Step 4-5: We apply \( \text{TC1} \) twice (since there are no applications of \( \text{TC2} \) that yield new information). In Step 4 we define the node 4 and the edge \( (1, b, 4) \) and in Step 5 the node 5 and the edge \( (2, a, 5) \); see Fig. A.1c.

Step 6: The node 6 and the edge \( (2, b, 6) \) are defined. Applying \( \text{TC2} \) to the node 6 and the relation \( (b^3, b) \in R \) leads to the definition of the edge \( (6, b, 2) \). The output of step 6 is \( \Gamma_7 \); see Fig. A.1d.

Step 7: We apply \( \text{TC1} \) and define the node 7 and the edges \( (3, b, 7) \); see Fig. A.1e.

Step 8: The node 8 and edge \( (4, a, 8) \) are defined in \( \text{TC1} \). Applying \( \text{TC2} \) leads to the definition of edges \( (7, b, 3), (8, b, 3), \) and \( (8, a, 4) \); see Fig. A.1f.

Step 9: We apply \( \text{TC1} \) and define the node 9 and the edge \( (4, b, 9) \). We apply \( \text{TC2} \) to 4 and the relation \( (b^3, b) \) and define the edge \( (9, b, 4) \); see Fig. A.1g.

Step 10: We apply \( \text{TC1} \) and define the node 10 and the edge \( (5, a, 10) \). We apply \( \text{TC2} \) to the node 5 and the relation \( (a^3, a) \) and define the edge \( (10, a, 5) \); see Fig. A.1h.

Step 11-12: We apply \( \text{TC1} \) twice, since there are no applications of \( \text{TC2} \) yielding any new information. The nodes 11 and 12 and the edges \( (5, b, 11) \) and \( (6, a, 12) \) are defined; see Fig. A.1i.

Step 13: We apply \( \text{TC1} \) and define the node 13 and the edge \( (7, a, 13) \). We apply \( \text{TC2} \) to the node 1 and the relation \( ((ab)^2, a^2) \) and define the edge \( (13, b, 1) \). We also apply \( \text{TC2} \) to node 7 and the relation \( ((ab)^2, a^2) \) and define the edge \( (13, a, 7) \). Next, we apply \( \text{TC2} \) to the node 1 and the relation \( (b^3, b) \) resulting in the pair \( (1, 9) \) begin added to \( \kappa_{14} \). Finally, we apply \( \text{TC2} \) to the node 13 and the relation \( ((ab)^2, a^2) \) which yields \( (4, 13) \) being added to \( \kappa_{14} \); see Fig. A.1j.

Step 14: In this step \( \kappa_{14} \) is the least equivalence relation containing \{\( (1, 9), (4, 13) \)\}. Taking the quotient gives graph \( \Gamma_{15} \). In \( \Gamma_{15} \) there exist two edges \( (4, a, 7) \) and \( (4, a, 8) \) with the same source and label, and so the pair \( (7, 8) \) is added to \( \kappa_{15} \); see Fig. A.2a.

Step 15: Taking the quotient of \( \Gamma_{15} \) in \( \text{F2} \) by \( \kappa_{15} \) yields the graph in Fig. A.2b.

Step 16: We apply \( \text{TC1} \) and define the node 14 and edge \( (10, b, 14) \); see Fig. A.2c.

Step 17: We apply \( \text{TC1} \) and define the node 15 and the edge \( (11, a, 15) \). Applying \( \text{TC2} \) to 2 and the relation \( ((ab)^2, a^2) \) leads to the definition of the edge \( (15, b, 10) \) and applying \( \text{TC2} \) to 11 and the relation \( ((ab)^2, a^2) \) leads to the definition of edge \( (15, a, 11) \). Finally, an application of \( \text{TC2} \) to 15 and the relation \( (b^3, b) \) leads to the definition of the node \( (14, b, 10) \); see Fig. A.2d.

Step 18: The node 16 and the edge \( (11, b, 16) \) are defined. After an application of \( \text{TC2} \) to 5 and \( (b^3, b) \) leads to the definition of \( (16, b, 11) \); see Fig. A.2e.
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|----|----|
| ε | a | b | a² | ab | ba | b² | a²b | aba | ab² | ba² | bab |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| b²a | a²ba | ba²b | (ba)² | bab² | b²a² | b²ab | ba²ba | b²a²b | b²aba | b²ab² | b²a²ba |

Table A.1: A word labelling a path from 0 to each node in Example A.1; see Fig. A.1, Fig. A.2, Fig. A.3, Fig. A.4, and Fig. A.5.

**Step 19:** The node 17 and the edge (12, a, 17) are defined. An application of $\text{TC2}$ to 6 and $(b³, b)$ leads to the definition of $(17, a, 12)$; see Fig. A.2f.

**Step 20:** We apply $\text{TC1}$ and the node 18 and the edge $(12, b, 18)$ are defined; see Fig. A.3a.

**Step 21:** We apply $\text{TC1}$ and the node 19 and the edge $(14, a, 19)$ are defined. Applying $\text{TC2}$ to 5 and the relation $((ab)², a²)$ leads to the definition of the edge $(19, b, 5)$ and applying $\text{TC2}$ to 14 and the relation $((ab)², a²)$ leads to the definition of edge $(19, a, 14)$. We apply $\text{TC2}$ to 19 and $(b³, b)$ and to node 19 and $((ab)², a²)$ yielding the pairs $(5, 16)$ and $(11, 19)$ are added to $κ_{22}$; see Fig. A.3b.

**Step 22:** Taking the quotient of $Γ_{22}$ by $κ_{22}$ gives the graph $Γ_{23}$. In $Γ_{23}$ there are the edges $(11, a, 14)$ and $(11, a, 15)$ with source 11 and label $a$ and hence the pair $(14, 15)$ is added to $κ_{23}$; see Fig. A.3c.

**Step 23:** Taking the quotient gives graph $Γ_{24}$; see Fig. A.3d.

**Step 24:** We apply $\text{TC1}$ so that the node 20 and the edge $(17, b, 20)$ are defined; see Fig. A.4a.

**Step 25:** We apply $\text{TC1}$ and the node 21 and the edge $(18, a, 21)$ are defined. We apply $\text{TC2}$ to 6 and $((ab)², a²)$ and define the edge $(21, b, 17)$. We also apply $\text{TC2}$ to 18 and $((ab)², a²)$ and define $(21, a, 18)$. Finally, we apply $\text{TC2}$ to 6 and $(b³, b)$ and we define $(20, b, 17)$; see Fig. A.4b.

**Step 26:** We apply $\text{TC1}$ and define 22 and $(18, b, 22)$. Next, we apply $\text{TC2}$ to 12 and $(b³, b)$. This leads to the definition of $(22, b, 18)$; see Fig. A.4c.

**Step 27:** We apply $\text{TC1}$ and define 23 and $(20, a, 23)$. We apply $\text{TC2}$ to 12 and $((ab)², a²)$ and define $(23, b, 12)$. We also apply $\text{TC2}$ to 20 and $((ab)², a²)$ and define $(23, a, 20)$. Finally, we apply $\text{TC2}$ to 23 and $(b³, b)$ and to 23 and $((ab)², a²)$ yielding the pairs $(12, 22)$ and $(18, 23)$ in $κ_{28}$; see Fig. A.4d.

**Step 28:** Taking the quotient of $Γ_{28}$ by $κ_{28}$ gives $Γ_{29}$. In $Γ_{29}$ there exist two edges $(18, a, 20)$ and $(18, a, 21)$ labelled $a$ leaving $b²ab$ and hence the pair $(20, 21)$ is added to $κ_{29}$; see Fig. A.5a.

**Step 29:** Taking the quotient of $Γ_{29}$ by $κ_{29}$ gives the graph $Γ_{30}$; see Fig. A.5b.

After Step 29, $Γ_{30}$ is complete, deterministic, and compatible with $R$. Hence the enumeration terminates.

**Example A.2.** In this example, we perform the HLT strategy to find the right Cayley graph of the monoid defined by the presentation:

$$(a, b, c \mid a² = ac, \ b² = b, \ a² = ca, \ bc = cb, \ a³ = c², \ a² = a², \ b² = ab)$$

with $a < b < c$. Each step of this enumeration corresponds to either: at least one application of $\text{TC1}$ (in $\text{HLT1}$) followed by multiple applications of $\text{TC2}$ (in the same $\text{HLT1}$ step as the application of $\text{TC1}$ or subsequent applications of $\text{HLT1}$ where $\text{TC1}$ is not invoked); or a single application of $\text{TC3}$ (in $\text{HLT2}$); see Fig. A.6 and Fig. A.7 for the word graphs $Γ_i$ and equivalence relation $κ_i$ after every step $i$; see also Table A.2.

**Step 1:** At this step we apply $\text{HLT1}$ to the node 0 and relation $(a², ac)$. $\text{TC1}$ yields the new nodes 1 and 2, and the edges $(0, a, 1)$ and $(1, a, 2)$; $\text{TC2}(b)$ then yields the edge $(1, c, 2)$; see Fig. A.6a.

**Step 2:** We apply $\text{HLT1}$ to the node 0 and relation $(b², b)$. $\text{TC1}$ produces the new node 3 and edge $(0, b, 3)$; $\text{TC2}(a)$ then yields the edge $(3, b, 3)$; see Fig. A.6b.

**Step 3:** $\text{HLT1}$ is applied to 0 and $(a², ca)$. $\text{TC1}$ generates the node 4 and edge $(0, c, 4)$; $\text{TC2}(b)$ gives us the edge $(4, a, 2)$; see Fig. A.6c.
Figure A.1: The output \((\Gamma_i, \kappa_i)\) for \(i = \{3, 4, 6, 7, 8, 9, 10, 11, 13, 14\}\) in the Felsch Strategy in Example 7.1. Purple arrows correspond to \(a\), gray to \(b\), shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in \(TC1\), a dashed edge with a single arrowhead denotes a new edge obtained from \(TC2\) or \(TC3\), solid edges correspond to edges that existed at the previous step.
(a) Step 14.

(b) Step 15.

(c) Step 16.

(d) Step 17.

(e) Step 18.

(f) Step 19.

Figure A.2: The output \((\Gamma_i, \kappa_i)\) for \(i \in \{15, \ldots, 20\}\) in the Felsch strategy in Example 7.1. Purple arrows correspond to \(a\), gray to \(b\), shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in \(F1\), a dashed edge with a single arrowhead denotes an edge that is obtained from \(F2\), solid edges correspond to edges that existed at the previous step.
Figure A.3: The output $(\Gamma_i, \kappa_i)$ for $i \in \{21, 22, 23, 24\}$ in the Felsch strategy in Example 7.1. Purple arrows correspond to $a$, gray to $b$, shaded nodes of the same colour belong to $\kappa_i$, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in $\text{TC1}$, a dashed edge with a single arrowhead denotes an edge that is obtained from $\text{TC2}$ or $\text{TC3}$, solid edges correspond to edges that existed at the previous step.
Figure A.4: The output \((\Gamma_i, \kappa_i)\) for \(i \in \{25, \ldots, 28\}\) in the Felsch strategy for Example 7.1. Purple arrows correspond to \(a\), gray to \(b\), shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in TC1, a dashed edge with a single arrowhead denotes an edge that is obtained from TC2 or TC3, solid edges correspond to edges that existed at the previous step.
Figure A.5: The output ($\Gamma_i, \kappa_i$) for $i = 29$ and 30 in the Felsch strategy of Example 7.1. Purple arrows correspond to $a$, gray to $b$, shaded nodes of the same colour belong to $\kappa_i$, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in $\textbf{TC}1$, a dashed edge with a single arrowhead denotes an edge that is obtained from $\textbf{TC}2$ or $\textbf{TC}3$, solid edges correspond to edges that existed at the previous step.

**Steps 4-6**: Step 4 is $\textbf{HLT1}$ applied to 0 and $(bc, cb)$; $\textbf{TC1}$ gives the new node 5 and edge $(3, c, 5)$; $\textbf{TC2}(b)$ yields the edge $(4, b, 5)$. Step 5 is $\textbf{HLT1}$ applied to 0 and $(a^2, c^2)$. There is already a path from 0 labelled by $a^2$ and another by $c$, and so $\textbf{TC1}$ is not invoked, but $\textbf{TC2}(b)$ yields the edge $(4, c, 2)$. Step 6: $\textbf{HLT1}$ applied to 0 and $(a^2, a^2)$, $\textbf{TC1}$ is not invoked again, and $\textbf{TC2}(a)$ yields the edge $(2, a, 2)$; see Fig. A.6d.

**Steps 7-9**: Step 7: $\textbf{HLT1}$ applied to 0 and $(a^2, aba)$; $\textbf{TC1}$ adds the node 6 and edge $(1, b, 6)$; $\textbf{TC2}(b)$ yields $(6, a, 2)$. Step 8: $\textbf{HLT1}$ applied to 1 and $(a^2, ac)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(b)$ yields $(2, c, 2)$. Step 9: $\textbf{HLT1}$ applied to 1 and $(b^2, b)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(a)$ yields $(6, b, 6)$. See Fig. A.6e.

**Steps 10-13**: Step 10: $\textbf{HLT1}$ applied to 1 and $(bc, cb)$; $\textbf{TC1}$ adds node 7 and edge $(6, c, 7)$. Step 11: $\textbf{HLT1}$ applied to 1 and $(a^2, aba)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(b)$ yields edge $(7, a, 2)$. Step 12: $\textbf{HLT1}$ applied to 2 and $(b^2, b)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(a)$ yields edge $(7, b, 7)$. Step 13: $\textbf{HLT1}$ applied to 2 and $(bc, cb)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(a)$ yields edge $(7, c, 7)$. See Fig. A.6f.

**Steps 14-18**: Step 14: $\textbf{HLT1}$ applied to 3 and $(a^2, ac)$; $\textbf{TC1}$ yields nodes 8 and 9 and edges $(3, a, 8), (8, a, 9)$; $\textbf{TC2}(b)$ yields edge $(8, c, 9)$. Step 15: $\textbf{HLT1}$ for 3 and $(a^2, ca)$; $\textbf{TC1}$ not applied; $\textbf{TC2}(b)$ yields edge $(5, a, 9)$. Step 16: $\textbf{HLT1}$ for $b$ and $(bc, cb)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(b)$ yields $(5, b, 5)$. Step 17: $\textbf{HLT1}$ for $b$ and $(a^2, c^2)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(b)$ yields $(5, c, 9)$. Step 18: $\textbf{HLT1}$ for 3 and $(a^3, a^2)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(a)$ yields $(9, a, 9)$. See Fig. A.6g.

**Steps 19-20**: Step 19: $\textbf{HLT1}$ for 3 and $(a^2, aba)$; $\textbf{TC1}$ adds node 10 and edge $(8, b, 10)$; $\textbf{TC2}(b)$ yields $(10, a, 9)$. Step 20: $\textbf{HLT1}$ for 4 and $(bc, cb)$; $\textbf{TC1}$ does not apply; $\textbf{TC2}(c)$ indicates $\kappa_{20}$ is the least equivalence containing $(7, 9)$. See Fig. A.6h.

**Step 21**: This step is an application of $\textbf{TC3}$ to produce $\Gamma_{21} := \Gamma_{20}/\kappa_{20}$ resulting in the new edges $(7, a, 7), (5, a, 7), (5, c, 7), (8, a, 7), (8, c, 7),$ and $(10, b, 7)$. The edge $(7, a, 2)$ is also an edge in $\Gamma_{21}$ meaning that $\kappa_{21}$ is the least equivalence containing $(2, 7)$. See Fig. A.7a.

**Step 22**: This step is also an application of $\textbf{TC3}$ to produce $\Gamma_{22} := \Gamma_{21}/\kappa_{21}$, the new edges created are $(2, b, 2), (10, a, 2), (8, a, 2), (8, c, 2), (5, a, 2),$ and $(5, c, 2)$. See Fig. A.7b.
Table A.2: A word labelling a path from 0 to each node in the right Cayley graph on the monoid $M$ from Example A.2; see Fig. A.6 and Fig. A.7.

Steps 23-24: Step 23: HLT1 applied to $(b^2, b)$ from 8; TC1 does not apply; TC2(a) yields $(10, b, 10)$. Step 24: HLT1 applied to 8 and $(bc, cb)$ yields $(10, c, 2)$. See Fig. A.7c.

After Step 24 the graph $\Gamma_{25}$ is complete and compatible with the relations in the presentation, and $\kappa_{25}$ is trivial, the enumeration terminates, and we see that the semigroup defined by the presentation is isomorphic to that defined in Example 5.1.

Example A.3. In this example, we perform the Felsch strategy to find the right Cayley graph of the monoid defined by the presentation:

$$\langle a, b, c \mid ac = a^2, b^2 = b, ca = a^2, cb = bc, a^2 = a^2, a^3 = a^2, aba = a^2 \rangle.$$

Step 1: At this step we apply $F1$ to the node 0. The node 1 and the edge $(0, a, 1)$ are defined; see Fig. A.8a.

Step 2: We apply $F1$ to the node 0. The node 2 and the edge $(0, b, 2)$ are defined. Applying $F2$ to 0 and the relation $(b^2, b)$ leads to the definition of the edge $(2, b, 2)$; see Fig. A.8b.

Step 3: We apply $F1$ to 0. The node 3 and the edge $(0, c, 3)$ are defined; see Fig. A.8c.

Step 4: $F1$ is applied to 1 and hence the node 4 and the edge $(1, a, 4)$ are defined. We apply $F2$ and the following applications of TC2 yield new information; applying TC2(b) to 0 and $(a^2, ac)$ yields the edge $(1, c, 4)$, applying TC2(b) to 0 and $(a^2, cb)$ yields the edge $(3, b, 4)$, the application of TC2(b) to 0 and $(a^2, c^2)$ yields the edge $(3, c, 4)$, the application of TC2(a) to 0 and $(a^3, a^2)$ yields the edge $(4, a, 4)$ and finally the application of TC2(b) to 1 and $(a^2, ac)$ yields the edge $(4, c, 4)$; see Fig. A.8d.

Step 5: We apply $F1$ to 1 which leads to the definition of the node 5 and the edge $(1, b, 5)$. We apply $F2$ and the following applications of TC2 yield new information; the application of TC2(b) to 0 and $(a^2, aba)$ yields the edge $(5, a, 4)$ and the application of TC2(a) to 1 and $(b^2, b)$ yields the edge $(5, b, 5)$; see Fig. A.8e.

Step 6: We apply $F1$ to 2 which leads to the definition of the node 6 and the edge $(2, a, 6)$; see Fig. A.8f.

Step 7: We apply $F1$ to 2 which leads to the definition of the node 7 and the edge $(2, c, 7)$. We apply $F2$ and the following applications of TC2 yield new information; applying TC2(b) to 0 and $(bc, cb)$ yields the edge $(3, b, 7)$ and applying TC2(b) to 3 and $(bc, cb)$ yields the edge $(7, b, 7)$; see Fig. A.9a.

Step 8: We apply $F1$ and the node 8 and the edge $(4, b, 8)$ are defined. We apply $F2$ and the following applications of TC2 yield new information; applying TC2(a) to 1 and $(bc, cb)$ yields the edge $(5, c, 8)$, applying TC2(b) to 1 and $(a^2, aba)$ yields the edge $(8, a, 4)$, applying TC2(a) to 3 and $(bc, cb)$ yields the edge $(7, c, 8)$, applying TC2(a) to 4 and $(b^2, b)$ yields the edge $(8, b, 8)$ and applying TC2(a) to 4 and $(bc, cb)$ yields the edge $(8, c, 8)$. In addition, applying TC2(c) to 5 and $(a^2, c^2)$ as well as applying TC2(c) to 8 and $(a^2, c^2)$ indicates $\kappa_9$ is the least equivalence containing $(8, 4)$ and $\Delta_{N_6}$; see Fig. A.9b.

Step 9: This step is an application of TC3 to produce $\Gamma_{10} := \Gamma_9/\kappa_9$ resulting in the new edges $(5, c, 4), (4, b, 4)$; see Fig. A.9c.

Step 10: We apply $F1$. The node 9 and the edge $(6, a, 9)$ are defined and an application of $F2$ follows. Applying TC2(b) to 2 and $(a^2, ac)$ yields the edge $(6, c, 9)$, applying TC2(b) to 2 and $(a^2, ca)$ yields the edge $(7, a, 9)$, applying TC2(a) to 2 and $(a^3, a^2)$ yields the edge $(9, a, 9)$ and applying TC2(b) to 6 and $(a^2, ac)$ yields the edge $(9, c, 9)$. In addition, applying TC2(c) to 2 and $(a^2, c^2)$ indicates $\kappa_{11}$ is the least equivalence containing $(4, 9)$ and $\Delta_{N_{11}}$; see Fig. A.9d.

Step 11: This step is an application of TC3 to produce $\Gamma_{12} := \Gamma_{11}/\kappa_{11}$ resulting in the new edges $(7, a, 4), (6, a, 4), (6, c, 4)$; see Fig. A.9e.

Step 12: We apply $F1$ and the node 10 and the edge $(6, b, 10)$ are defined. We apply $F2$ and the following applications of TC2 yield new information; applying TC2(b) to 2 and $(a^2, aba)$ yields the edge $(10, a, 4)$, applying TC2(a) to 6 and $(b^2, b)$ yields the edge $(10, b, 10)$ and applying TC2(a) to 6 and $(bc, cb)$ yields the edge $(10, c, 4)$; see Fig. A.9f.
Figure A.6: The output \((\Gamma_i, \kappa_i)\) for \(i = 1, \ldots, 20\) in Example A.2. Purple arrows correspond to \(a\), gray to \(b\), pink to \(c\), shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in TC1, a dashed edge with a single arrowhead denotes an edge that is obtained from TC2 or TC3, solid edges correspond to edges that existed at the previous step.
Figure A.7: The output \((\Gamma_i, \kappa_i)\) for \(i = 21, \ldots, 24\) of each step in Example A.2. Purple arrows correspond to 1, gray to 3, pink to 4, shaded nodes of the same colour belong to \(\kappa_i\), and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in TC1, a dashed edge with a single arrowhead denotes an edge that is obtained from TC2 or TC3, solid edges correspond to edges that existed at the previous step.
Table A.3: A word labelling a path from 0 to each node in the right Cayley graph on the monoid $M$ from Example A.3; see Fig. A.8 and Fig. A.9.

(a) Step 1.

(b) Step 2.

(c) Step 3.

(d) Step 4.

(e) Step 5.

(f) Step 6.

Figure A.8: The output $(\Gamma_i, \kappa_i)$ of each step in Example A.3. Purple arrows correspond to $a$, grey to $b$, pink to $c$, shaded nodes of the same colour belong to $\kappa_i$, and unshaded nodes belong to singleton classes.

After Step 12 the graph $\Gamma_{13}$ is complete and compatible with the relations in the presentation, and $\kappa_{13}$ is trivial, the enumeration terminates, and we see that the semigroup defined by the presentation is isomorphic to that defined in Example 5.1.

B Performance comparison

In this appendix the performance of the libsemigroups [24] implementations of the HLT and Felsch strategies (as described in this paper) is compared to the performance of GAP [12] on a number of examples. Unless otherwise indicated the times for libsemigroups [24] are the mean time of 100 runs and the indicated times for GAP [12] are the mean of 10 runs. All computations were performed on a 2021 Mac M1 computer with 16GB of RAM.

B.1 Monoids of transformations

In this section we present some tables of timings for some presentations from the literature of transformation monoids; see Tables B.1 and B.2.

B.2 Monoids of partitions

In this section we present some tables of timings for some presentations from the literature of monoids of partitions; see Tables B.3, B.4, B.5, B.6, and B.7.
Figure A.9: The output $(\Gamma_i, \kappa_i)$ of each step in Example A.3. Purple arrows correspond to $a$, grey to $b$, pink to $c$, shaded nodes of the same colour belong to $\kappa_i$, and unshaded nodes belong to singleton classes.
B.3 Further finitely presented semigroups and monoids

In this section we present some tables of timings for some presentations from the literature of finitely presented semigroups and monoids; see Tables B.8, B.9, and B.10.

The presentations referred to in Table B.10 are:

\[
S = \langle a, b, c \mid a^{14} = a, b^{14} = b, c^{14} = c, a^4ba = b^3, b^4ab = a^3, a^4ca = c^3, c^4ac = a^3, b^4cb = c^3, c^4bc = b^3 \rangle \quad (B.1)
\]

\[
S = \langle a, b \mid a^{32} = a, b^3 = b, abab = b, a^{16}ba^4ba^4 \rangle \quad (B.2)
\]

\[
S = \langle a, b \mid a^{16} = a, b^6 = b, ab^2 = ba^2 \rangle \quad (B.3)
\]

\[
S = \langle a, b \mid a^{3} = a, b^6 = b, (abab)^7ab^2a = b^2 \rangle \quad (B.4)
\]

\[
S = \langle a, b \mid a^{3} = a, b^6 = b, (abab)^7ab^2a^2b = b^2 \rangle \quad (B.5)
\]

\[
S = \langle a, b, c, d, e \mid a^{3} = a, b^3 = b, c^3 = c, d^3 = d, e^3 = 3, (ab)^3 = a^2, (be)^3 = b^2, (cd)^3 = c^2, (de)^3 = d^2, \nonumber
\]

\[
ac = ca, ad = da, ae = ea, bd = db, be = eb, ce = ec
\]

\[
S = \langle a, b \mid a^3 = a, b^{23} = b, ab^{11}ab^2 = b^2a \rangle \quad (B.6)
\]

B.4 Finitely presented groups

In this section we provide some comparison of the performance of libsemigroups [24], GAP [12], and ACE [14] when applied to group presentations. It might be worth noting that libsemigroups [24] contains no optimizations whatsoever for group presentations, and that when run in libsemigroups [24] each of the presentations given in this section has: an explicit generator \(a^{-1}\) for the inverse of each generator \(a\) that is not of order 2; the relations \(aa^{-1} = a^{-1}a = e\) for every generator \(a\). The version of Todd-Coxeter in GAP [12] is largely written in C, and is specific to group presentations; similarly, ACE [14] is written in C and is specific to groups also. As such, it is not surprising that the performance of libsemigroups [24] is generally worse that both ACE [14] and GAP [12] when applied to a group presentation. There are still some cases where libsemigroups [24] is faster than GAP [12]; see Table B.11.

\[
G = \langle a, b, c \mid ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, ac^2a^{-1}a^{-1}b^{-1}ab \rangle \quad (B.9)
\]

\[
H = \langle bc \rangle \quad (B.10)
\]

\[
G = \langle a, b, c \mid ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, ac^2a^{-1}a^{-1}b^{-1}ab \rangle \quad (B.11)
\]

\[
H = \langle bc, a^{-1}b^{-1}(a^{-1})^2bcabc^{-1} \rangle \quad (B.12)
\]

\[
G = \langle a, b, c \mid ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, ac^2a^{-1}a^{-1}b^{-1}ab \rangle \quad (B.13)
\]

\[
H = \langle bc, a^{-1}b^{-1}a^{-1}b^{-1}a^{-1}b^{-1}ab \rangle \quad (B.14)
\]

\[
G = F_{2,7} = \langle a, b, c, d, x, y, z \mid a^{-1}bc, bcd^{-1}, cdx^{-1}, dxy^{-1}, yza^{-1}, zab^{-1} \rangle \quad (B.15)
\]

\[
H = \langle e \rangle \quad (B.16)
\]

\[
G = M_{12} = \langle a, b, c \mid a^{11}, b^2, c^2, (ab)^3, (ac)^3, (bc)^{10}, (ab)^2abcabc(a^{-1})^5 \rangle \quad (B.17)
\]

\[
H = \langle e \rangle \quad (B.18)
\]

\[
G = SL(2, 19) = \langle a, b \mid ab^{-1}a^{-1}b^{-1}a^{-1}b^{-1}, b^{-1}a^{-1}a^{-1}b^{-1}, ba^4b^{-1}a^4b^{-1}ab^{-1} \rangle \quad (B.19)
\]

\[
H = \langle b \rangle \quad (B.20)
\]

\[
G = \langle a, b, c, x \mid x^2, y^3, ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, acca^{-1}a^{-1}b^{-1}ab, a^{-1}xax, b^{-1}xbx, c^{-1}xcx \rangle \quad (B.21)
\]

\[
H = \langle a^{-1}y^{-1}ay, b^{-1}y^{-1}by, c^{-1}y^{-1}cy, xy^{-1}xy \rangle \quad (B.22)
\]

\[
G = \langle ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, ac^2a^{-2}b^{-1}ab \rangle \quad (B.23)
\]
Table B.1: Comparison of libsemigroups [24] and the implementation in the GAP [12] library (CosetTableOfFpSemigroup) for the presentations of the monoids $OP_n$ of orientation preserving transformations of a chain from [2]. Values with no standard deviation indicated were only run once. All times are in seconds.

| n  | $|OP_n|$ | $|A|$ | $|R|$ | $\sum |uv|$ | HLT mean | s.d. | Felsch mean | s.d. | GAP mean | s.d. |
|----|--------|-----|-----|--------|---------|-----|-------------|-----|---------|-----|
| 3  | 24     | 2   | 5   | 43     | $8.4 \times 10^{-6}$ | $8 \times 10^{-7}$ | $1.84 \times 10^{-5}$ | $1.8 \times 10^{-6}$ | $5.3 \times 10^{-4}$ | $7 \times 10^{-5}$ |
| 4  | 128    | 2   | 6   | 86     | $4.0 \times 10^{-5}$ | $2 \times 10^{-6}$ | $2.77 \times 10^{-4}$ | $5 \times 10^{-6}$ | $1.92 \times 10^{-3}$ | $7 \times 10^{-5}$ |
| 5  | 610    | 2   | 7   | 145    | $4.51 \times 10^{-4}$ | $4 \times 10^{-6}$ | $4.565 \times 10^{-3}$ | $1.5 \times 10^{-5}$ | $1.492 \times 10^{-2}$ | $1.3 \times 10^{-4}$ |
| 6  | 2,742  | 2   | 8   | 220    | $4.64 \times 10^{-3}$ | $7 \times 10^{-5}$ | $4.153 \times 10^{-2}$ | $1.2 \times 10^{-4}$ | $1.174 \times 10^{-1}$ | $3 \times 10^{-4}$ |
| 7  | 11,970 | 2   | 9   | 311    | $4.97 \times 10^{-2}$ | $4 \times 10^{-4}$ | $4.0 \times 10^{-1}$ | $5 \times 10^{-2}$ | $8.081 \times 10^{-1}$ | $1.1 \times 10^{-3}$ |
| 8  | 51,424 | 2   | 10  | 418    | $7.11 \times 10^{-1}$ | $5 \times 10^{-3}$ | $3.07 \times 10^{0}$ | $4 \times 10^{-2}$ | $4.88 \times 10^{0}$ | $2 \times 10^{-3}$ |
| 9  | 218,718| 2   | 11  | 541    | $5.40 \times 10^{0}$ | $3 \times 10^{-2}$ | $2.98 \times 10^{1}$ | $2 \times 10^{-1}$ | -          | -    |
| 10 | 923,690| 2   | 12  | 680    | $2.503 \times 10^{1}$ | -           | -          | -          | -          | -    |
| 11 | 3,879,766| 2   | 13  | 835    | $2.680 \times 10^{2}$ | -          | -          | -          | -          | -    |
| 12 | 16,224,804| 2   | 14  | 1,006  | $2.351 \times 10^{3}$ | -          | -          | -          | -          | -    |
| 13 | 67,603,744| 2   | 15  | 1,193  | $3.160 \times 10^{4}$ | -          | -          | -          | -          | -    |

Table B.2: Comparison of libsemigroups [24] and the implementation in the GAP [12] library (CosetTableOfFpSemigroup) for the presentations of the monoids $OR_n$ of orientation preserving and reversing transformations of a chain from [2]. Values with no standard deviation indicated were only run once. All times are seconds.

| n  | $|OR_n|$ | $|A|$ | $|R|$ | $\sum |uv|$ | HLT mean | s.d. | Felsch mean | s.d. | GAP mean | s.d. |
|----|--------|-----|-----|--------|---------|-----|-------------|-----|---------|-----|
| 3  | 27     | 3   | 9   | 65     | $1.388 \times 10^{-5}$ | $9 \times 10^{-8}$ | $3.75 \times 10^{-5}$ | $3 \times 10^{-7}$ | $6.78 \times 10^{-4}$ | $1.2 \times 10^{-5}$ |
| 4  | 180    | 3   | 10  | 122    | $8.45 \times 10^{-5}$ | $6 \times 10^{-7}$ | $5.784 \times 10^{-4}$ | $1.7 \times 10^{-6}$ | $4.18 \times 10^{-3}$ | $1.5 \times 10^{-4}$ |
| 5  | 2515   | 3   | 11  | 199    | $1.120 \times 10^{-3}$ | $1.0 \times 10^{-5}$ | $1.034 \times 10^{-2}$ | $5 \times 10^{-5}$ | $3.140 \times 10^{-2}$ | $8 \times 10^{-5}$ |
| 6  | 5,028  | 3   | 12  | 296    | $1.257 \times 10^{-2}$ | $6 \times 10^{-5}$ | $1.459 \times 10^{-1}$ | $3 \times 10^{-4}$ | $2.466 \times 10^{-1}$ | $2 \times 10^{-4}$ |
| 7  | 23,051 | 3   | 13  | 413    | $2.404 \times 10^{-1}$ | $1.2 \times 10^{-3}$ | $1.369 \times 10^{0}$ | $5 \times 10^{-3}$ | $1.8565 \times 10^{0}$ | $1.1 \times 10^{-3}$ |
| 8  | 101,272| 3   | 14  | 550    | $4.94 \times 10^{0}$ | $3 \times 10^{-2}$ | $1.437 \times 10^{1}$ | $4 \times 10^{-2}$ | $1.302 \times 10^{1}$ | $4 \times 10^{-2}$ |
| 9  | 434,835| 3   | 15  | 707    | $7.035 \times 10^{0}$ | -          | -          | -          | -          | -    |
| 10 | 1,843,320| 3   | 16  | 884    | $8.509 \times 10^{1}$ | -          | -          | -          | -          | -    |
| 11 | 7,753,471| 3   | 17  | 1,081  | $8.460 \times 10^{2}$ | -          | -          | -          | -          | -    |
| 12 | 32,440,884| 3   | 18  | 1,298  | $6.178 \times 10^{3}$ | -          | -          | -          | -          | -    |
Table B.3: Comparison of \textit{libsemigroups} \cite{24} and the implementation in the GAP \cite{12} library (\texttt{CosetTableOfFpSemigroup}) for the presentations of the partition monoids $P_n$ from \cite[Theorem 41]{8}. Values with no standard deviation indicated were only run once. All times are in seconds.

| $n$ | $|P_n|$ | $|A|$ | $|R|$ | $\sum |uv|$ | HLT mean | s.d. | Felsch mean | s.d. | GAP mean | s.d. |
|-----|--------|------|------|-------|-----------|------|------------|------|----------|------|
| 4   | 4,140  | 4    | 19   | 144   | $7.43 \times 10^{-3}$ | $1.2 \times 10^{-4}$ | $1.918 \times 10^{-2}$ | $5 \times 10^{-5}$ | $1.2919 \times 10^{-1}$ | $1.1 \times 10^{-4}$ |
| 5   | 115,975| 4    | 19   | 163   | $5.69 \times 10^{-1}$ | $2 \times 10^{-3}$ | $1.082 \times 10^{0}$ | $7 \times 10^{-3}$ | $5.377 \times 10^{0}$ | $6 \times 10^{-3}$ |
| 6   | 4,213,597| 4   | 20   | 198   | $3.24 \times 10^{1}$ | $3 \times 10^{-1}$ | $1.460 \times 10^{2}$ | $5 \times 10^{-1}$ | -         | -     |
| 7   | 190,899,322| 4   | 20   | 219   | $2.240 \times 10^{3}$ | -     | -           | -     | -         | -     |

Table B.4: Comparison of \textit{libsemigroups} \cite{24} and the implementation in the GAP \cite{12} library (\texttt{CosetTableOfFpSemigroup}) on the presentations for the singular Brauer monoids $B_n \setminus S_n$ from \cite{21}. Values with no standard deviation indicated were only run once. All times are in seconds.

| $n$ | $|B_n \setminus S_n|$ | $|A|$ | $|R|$ | $\sum |uv|$ | HLT mean | s.d. | Felsch mean | s.d. | GAP mean | s.d. |
|-----|----------------------|------|------|-------|-----------|------|------------|------|----------|------|
| 3   | 9                    | 6    | 21   | 78    | $1.72 \times 10^{-5}$ | $2 \times 10^{-7}$ | $2.4 \times 10^{-5}$ | $4 \times 10^{-6}$ | $4.6 \times 10^{-4}$ | $4 \times 10^{-5}$ |
| 4   | 81                   | 12   | 126  | 600   | $1.955 \times 10^{-4}$ | $8 \times 10^{-7}$ | $6.27 \times 10^{-4}$ | $1.8 \times 10^{-5}$ | $5.50 \times 10^{-3}$ | $1.2 \times 10^{-4}$ |
| 5   | 825                  | 20   | 450  | 2,300 | $5.33 \times 10^{-3}$ | $5 \times 10^{-5}$ | $1.968 \times 10^{-2}$ | $1.8 \times 10^{-4}$ | $1.60 \times 10^{-1}$ | $2 \times 10^{-3}$ |
| 6   | 9,675                | 30   | 1,185| 6,240 | $1.359 \times 10^{-1}$ | $2 \times 10^{-4}$ | $6.59 \times 10^{-1}$ | $6 \times 10^{-3}$ | $4.81 \times 10^{0}$ | $4 \times 10^{-2}$ |
| 7   | 130,095              | 42   | 2,583| 13,818| $3.89 \times 10^{0}$ | $2 \times 10^{-2}$ | $2.190 \times 10^{1}$ | $1.2 \times 10^{-1}$ | -         | -     |
| 8   | 1,986,705            | 56   | 4,956| 26,768| $3.357 \times 10^{2}$ | -     | -           | -     | -         | -     |
Table B.5: Comparison of libsemigroups [24] and the implementation in the GAP [12] library (CosetTableOfFpSemigroup) the presentations for the dual symmetric inverse monoids $I_n^*$ from [7]. This monoid is sometimes called the uniform block bijection monoid. Values with no standard deviation indicated were only run once. All times are in seconds.

| $n$ | $|I_n^*|$ | $|A|$ | $|R|$ | $\sum |uv|$ | HLT mean | HLT s.d. | Felsch mean | Felsch s.d. | Rc mean | Rc s.d. | GAP mean | GAP s.d. |
|-----|---------|-----|-----|--------|---------|---------|-----------|-----------|---------|---------|----------|---------|
| 3   | 25      | 3   | 11  | 54     | $1.5 \times 10^{-5}$ | $5 \times 10^{-6}$ | $1.92 \times 10^{-5}$ | $5 \times 10^{-7}$ | $1.85 \times 10^{-5}$ | $8 \times 10^{-8}$ | $4.94 \times 10^{-4}$ | $1.0 \times 10^{-5}$ |
| 4   | 339     | 4   | 19  | 116    | $4.17 \times 10^{-4}$ | $4 \times 10^{-6}$ | $6.05 \times 10^{-4}$ | $4 \times 10^{-6}$ | $1.10 \times 10^{-3}$ | $9 \times 10^{-6}$ | $8.74 \times 10^{-3}$ | $2 \times 10^{-5}$ |
| 5   | 6,721   | 5   | 28  | 177    | $3.15 \times 10^{-2}$ | $3 \times 10^{-4}$ | $2.78 \times 10^{-2}$ | $7 \times 10^{-5}$ | $2.92 \times 10^{-2}$ | $6 \times 10^{-5}$ | $3.21 \times 10^{-1}$ | $5 \times 10^{-4}$ |
| 6   | 179,643 | 6   | 39  | 276    | $3.63 \times 10^0$ | $9 \times 10^{-3}$ | $1.39 \times 10^0$ | $1.2 \times 10^{-2}$ | $4.04 \times 10^0$ | $1.3 \times 10^{-2}$ | $3.858 \times 10^1$ | $4 \times 10^{-2}$ |
| 7   | 6,166,105 | 7   | 52  | 422    | -       | -        | $1.115 \times 10^2$ | -         | -       | -       | -        | -        |

Table B.6: Comparison of libsemigroups [24] and the implementation in the GAP [12] library (CosetTableOfFpSemigroup) the presentations for the factorisable dual symmetric inverse monoids $FI_n^*$ from [10]. This monoid is sometimes called the uniform block bijection monoid. Values with no standard deviation indicated were only run once. All times are in seconds.

| $n$ | $|FI_n^*|$ | $|A|$ | $|R|$ | $\sum |uv|$ | HLT mean | HLT s.d. | Felsch mean | Felsch s.d. | Rc mean | Rc s.d. | GAP mean | GAP s.d. |
|-----|-----------|-----|-----|--------|---------|---------|-----------|-----------|---------|---------|----------|---------|
| 3   | 16        | 3   | 8   | 47     | $1.1 \times 10^{-5}$ | $4 \times 10^{-6}$ | $2.6 \times 10^{-5}$ | $4 \times 10^{-6}$ | $1.57 \times 10^{-5}$ | $1.4 \times 10^{-6}$ | $4.48 \times 10^{-4}$ | $1.7 \times 10^{-5}$ |
| 4   | 131       | 4   | 13  | 67     | $8.8 \times 10^{-5}$ | $4 \times 10^{-6}$ | $1.54 \times 10^{-4}$ | $1.1 \times 10^{-5}$ | $1.02 \times 10^{-4}$ | $4 \times 10^{-6}$ | $2.30 \times 10^{-3}$ | $1.3 \times 10^{-5}$ |
| 5   | 1,496     | 5   | 20  | 95     | $2.64 \times 10^{-3}$ | $5 \times 10^{-5}$ | $2.24 \times 10^{-3}$ | $2 \times 10^{-5}$ | $3.35 \times 10^{-3}$ | $1.0 \times 10^{-4}$ | $3.61 \times 10^{-2}$ | $5 \times 10^{-5}$ |
| 6   | 22,482    | 6   | 29  | 131    | $7.98 \times 10^{-2}$ | $4 \times 10^{-4}$ | $5.39 \times 10^{-2}$ | $4 \times 10^{-4}$ | $5.67 \times 10^{-2}$ | $2 \times 10^{-4}$ | $8.18 \times 10^{-1}$ | $1.3 \times 10^{-3}$ |
| 7   | 426,833   | 7   | 40  | 175    | $4.78 \times 10^0$ | $4 \times 10^{-2}$ | $1.608 \times 10^0$ | $1.3 \times 10^{-2}$ | $4.162 \times 10^0$ | $1.1 \times 10^{-2}$ | $1.466 \times 10^2$ | $1.0 \times 10^0$ |
| 8   | 9,934,563 | 8   | 53  | 227    | $8.030 \times 10^1$ | -        | $5.572 \times 10^1$ | -         | -       | -       | -        | -        |
Table B.7: Comparison of \textsc{libsemigroups} [24] and the implementation in the \textsc{GAP} [12] library (\textit{CosetTableOfFpSemigroup}) on the presentations for the Temperley-Lieb monoids $J_n$ from [9, Theorem 2.2]; the Temperley-Lieb monoid is also sometimes referred to as the \textit{Jones monoid} in the literature. Values with no standard deviation indicated were only run once. All times are in seconds.

| $n$ | $|J_n|$ | $|A|$ | $|R|$ | $\sum |uv|$ | \textbf{HLT} & \textbf{Felsch} & \textbf{GAP} |
|-----|-------|------|------|----------|-----------------|-----------------|-----------------|
|     |       |      |      |          | mean | s.d. | mean | s.d. | mean | s.d. |
| 3   | 5     | 2    | 4    | 14       | $4.87 \times 10^{-6}$ | $6 \times 10^{-8}$ | $5.9 \times 10^{-6}$ | $4 \times 10^{-7}$ | $2.69 \times 10^{-4}$ | $1.2 \times 10^{-5}$ |
| 4   | 14    | 3    | 8    | 29       | $8.34 \times 10^{-6}$ | $8 \times 10^{-8}$ | $1.132 \times 10^{-5}$ | $1 \times 10^{-7}$ | $3.26 \times 10^{-4}$ | $6 \times 10^{-6}$ |
| 5   | 42    | 4    | 13   | 48       | $1.7 \times 10^{-5}$ | $2 \times 10^{-7}$ | $2.686 \times 10^{-5}$ | $1.5 \times 10^{-7}$ | $5.39 \times 10^{-4}$ | $1.1 \times 10^{-5}$ |
| 6   | 132   | 5    | 19   | 71       | $4.18 \times 10^{-5}$ | $6 \times 10^{-7}$ | $9.51 \times 10^{-5}$ | $1 \times 10^{-6}$ | $1.464 \times 10^{-3}$ | $1.1 \times 10^{-5}$ |
| 7   | 429   | 6    | 26   | 98       | $1.463 \times 10^{-4}$ | $1.4 \times 10^{-6}$ | $4.5 \times 10^{-4}$ | $4 \times 10^{-6}$ | $5.3 \times 10^{-3}$ | $6 \times 10^{-5}$ |
| 8   | 1,430 | 7    | 34   | 129      | $5.93 \times 10^{-4}$ | $4 \times 10^{-6}$ | $2.248 \times 10^{-3}$ | $1.0 \times 10^{-5}$ | $2.274 \times 10^{-2}$ | $5 \times 10^{-5}$ |
| 9   | 4,862 | 8    | 43   | 164      | $2.456 \times 10^{-3}$ | $9 \times 10^{-6}$ | $1.034 \times 10^{-2}$ | $5 \times 10^{-5}$ | $9.753 \times 10^{-2}$ | $1.4 \times 10^{-4}$ |
| 10  | 16,796| 9    | 53   | 203      | $1.052 \times 10^{-2}$ | $4 \times 10^{-5}$ | $4.57 \times 10^{-2}$ | $4 \times 10^{-4}$ | $4.15 \times 10^{-1}$ | $3 \times 10^{-3}$ |
| 11  | 58,786| 10   | 64   | 246      | $4.238 \times 10^{-2}$ | $1.4 \times 10^{-4}$ | $1.966 \times 10^{-1}$ | $1.4 \times 10^{-3}$ | $1.789 \times 10^{0}$ | $9 \times 10^{-3}$ |
| 12  | 208,012| 11   | 76   | 293      | $1.805 \times 10^{-1}$ | $1.6 \times 10^{-3}$ | $8.52 \times 10^{-1}$ | $8 \times 10^{-3}$ | $8.73 \times 10^{0}$ | $6 \times 10^{-2}$ |
| 13  | 742,900| 12   | 89   | 344      | $7.791 \times 10^{-1}$ | $7 \times 10^{-4}$ | $3.63 \times 10^{0}$ | $3 \times 10^{-2}$ | $4.40 \times 10^{1}$ | $4 \times 10^{-1}$ |
| 14  | 2,674,440| 13   | 103  | 399      | $3.235 \times 10^{0}$ | $9 \times 10^{-3}$ | $1.517 \times 10^{1}$ | $1.1 \times 10^{-1}$ | - | - |
| 15  | 9,694,845| 14   | 118  | 458      | $1.424 \times 10^{1}$ | - | - | - | - |
| 16  | 35,357,670| 15   | 134  | 521      | $6.419 \times 10^{1}$ | - | - | - | - |

Table B.8: Comparison of \textsc{libsemigroups} [24] and the implementation in the \textsc{GAP} [12] library (\textit{CosetTableOfFpSemigroup}) for the presentations of the stellar monoids from [13]. Values with no standard deviation indicated were only run once. All times are in seconds.

| $n$ | Stellar($n$) | $|A|$ | $|R|$ | $\sum |uv|$ | \textbf{HLT} & \textbf{Felsch} & \textbf{GAP} |
|-----|-------------|------|------|----------|-----------------|-----------------|-----------------|
|     |             |      |      |          | mean | s.d. | mean | s.d. | mean | s.d. |
| 3   | 16          | 3    | 9    | 45       | $9.9 \times 10^{-6}$ | $1.2 \times 10^{-6}$ | $1.43 \times 10^{-5}$ | $3 \times 10^{-7}$ | $4.10 \times 10^{-4}$ | $1.2 \times 10^{-5}$ |
| 4   | 65          | 4    | 14   | 71       | $2.239 \times 10^{-5}$ | $1.8 \times 10^{-7}$ | $5.35 \times 10^{-5}$ | $4 \times 10^{-7}$ | $9.69 \times 10^{-4}$ | $1.4 \times 10^{-5}$ |
| 5   | 326         | 5    | 20   | 103      | $1.017 \times 10^{-4}$ | $8 \times 10^{-7}$ | $4.30 \times 10^{-4}$ | $5 \times 10^{-6}$ | $4.80 \times 10^{-3}$ | $2 \times 10^{-5}$ |
| 6   | 1,957       | 6    | 27   | 141      | $8.29 \times 10^{-4}$ | $4 \times 10^{-6}$ | $4.10 \times 10^{-3}$ | $3 \times 10^{-5}$ | $3.660 \times 10^{-2}$ | $5 \times 10^{-5}$ |
| 7   | 13,700      | 7    | 35   | 185      | $9.27 \times 10^{-3}$ | $5 \times 10^{-5}$ | $4.192 \times 10^{-2}$ | $1.2 \times 10^{-4}$ | $3.310 \times 10^{-1}$ | $3 \times 10^{-4}$ |
| 8   | 109,601     | 8    | 44   | 235      | $1.554 \times 10^{-1}$ | $6 \times 10^{-4}$ | $4.603 \times 10^{-1}$ | $1.6 \times 10^{-3}$ | $3.473 \times 10^{0}$ | $3 \times 10^{-3}$ |
| 9   | 986,410     | 9    | 54   | 291      | $9.67 \times 10^{0}$ | $1.4 \times 10^{-1}$ | $5.62 \times 10^{0}$ | $5 \times 10^{-2}$ | - | - |
| 10  | 9,864,101   | 10   | 65   | 353      | - | - | $7.337 \times 10^{1}$ | - | - |
| 11  | 108,505,112 | 11   | 77   | 421      | - | - | $1.018 \times 10^{3}$ | - | - |
Table B.9: Comparison of libsemigroups [24] and the implementation in the GAP [12] library (CosetTableOfFpSemigroup) for the presentations of the stylic monoids from [1]. Values with no standard deviation indicated were only run once. All times are in seconds.

| n  | |Stylic(n)| |A| |R| |∑|uw| |HLT mean| s.d.| Felsch mean| s.d.| GAP mean| s.d.|
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 15 | 3 | 11 | 57 | 1.07 × 10^{-5} | 1.1 × 10^{-6} | 1.767 × 10^{-5} | 9 × 10^{-8} | 4.7 × 10^{-4} | 8 × 10^{-5} |
| 4 | 52 | 4 | 24 | 132 | 2.97 × 10^{-5} | 2 × 10^{-7} | 8.64 × 10^{-5} | 7 × 10^{-7} | 1.07 × 10^{-3} | 7 × 10^{-5} |
| 5 | 203 | 5 | 45 | 255 | 1.309 × 10^{-4} | 8 × 10^{-7} | 6.69 × 10^{-4} | 5 × 10^{-6} | 4.39 × 10^{-3} | 7 × 10^{-5} |
| 6 | 877 | 6 | 76 | 438 | 7.98 × 10^{-4} | 4 × 10^{-6} | 4.926 × 10^{-3} | 1.2 × 10^{-5} | 2.722 × 10^{-2} | 4 × 10^{-5} |
| 7 | 4,140 | 7 | 119 | 693 | 5.52 × 10^{-3} | 3 × 10^{-5} | 3.594 × 10^{-2} | 1.8 × 10^{-4} | 1.899 × 10^{-1} | 3 × 10^{-4} |
| 8 | 21,147 | 8 | 176 | 1,032 | 4.033 × 10^{-2} | 8 × 10^{-5} | 2.673 × 10^{-1} | 8 × 10^{-4} | 1.4019 × 10^{0} | 1.9 × 10^{-3} |
| 9 | 115,975 | 9 | 249 | 1,467 | 3.048 × 10^{-1} | 4 × 10^{-4} | 2.041 × 10^{0} | 8 × 10^{-3} | 1.0803 × 10^{1} | 1.5 × 10^{-2} |
| 10 | 678,570 | 10 | 340 | 2,010 | 2.5358 × 10^{0} | 1.4 × 10^{-3} | 1.625 × 10^{1} | 9 × 10^{-2} | - | - |
| 11 | 4,213,597 | 11 | 451 | 2,673 | 2.155 × 10^{1} | - | - | - | - |
| 12 | 27,644,437 | 12 | 584 | 3,468 | 2.503 × 10^{2} | - | - | - | - |

Table B.10: Comparison of libsemigroups [24] and the implementation in the GAP [12] library (CosetTableOfFpSemigroup) on examples from Walker [36]. All times are in seconds.

| n  | |S| |A| |R| |∑|uw| |HLT mean| s.d.| Felsch mean| s.d.| GAP mean| s.d.|
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (B.1) | 1 | 3 | 9 | 99 | 2.14 × 10^{-3} | 1 × 10^{-4} | 5.68 × 10^{-2} | 3 × 10^{-4} | 1.62 × 10^{-1} | 3 × 10^{-3} |
| (B.2) | 14,911 | 4 | 6 | 31 | 5.32 × 10^{-2} | 3 × 10^{-4} | 3.344 × 10^{-2} | 1.6 × 10^{-4} | 6.93 × 10^{0} | 5 × 10^{-2} |
| (B.3) | 20,490 | 2 | 3 | 40 | 2.454 × 10^{-1} | 5 × 10^{-4} | 2.382 × 10^{-2} | 1.0 × 10^{-4} | 5.19 × 10^{0} | 3 × 10^{-2} |
| (B.4) | 36,412 | 5 | 6 | 31 | 2.903 × 10^{-1} | 9 × 10^{-4} | 8.875 × 10^{-1} | 7 × 10^{-4} | 7.530 × 10^{0} | 1.6 × 10^{-2} |
| (B.5) | 72,822 | 5 | 6 | 34 | 4.708 × 10^{-1} | 1.9 × 10^{-3} | - | - | 2.401 × 10^{1} | 9 × 10^{-2} |
| (B.6) | 78,722 | 6 | 7 | 32 | 1.327 × 10^{-1} | 5 × 10^{-4} | 3.297 × 10^{0} | 1.8 × 10^{-2} | 6.922 × 10^{0} | 9 × 10^{-3} |
| (B.7) | 153,500 | 5 | 15 | 76 | 1.799 × 10^{-1} | 6 × 10^{-4} | 1.819 × 10^{-1} | 1.7 × 10^{-3} | 2.496 × 10^{0} | 9 × 10^{-3} |
| (B.8) | 270,272 | 2 | 3 | 46 | 8.002 × 10^{-1} | 1.9 × 10^{-3} | 2.67 × 10^{0} | 3 × 10^{-2} | 1.234 × 10^{1} | 2 × 10^{-2} |

Table B.11: Comparison of libsemigroups [24], ACE [14], and the implementation in the GAP [12] library (CosetTableFromGensAndRelS). All times are in seconds.