Semigroup Forum
https://doi.org/10.1007/s00233-024-10431-z

RESEARCH ARTICLE

The Todd–Coxeter algorithm for semigroups and monoids

T. D. H. Coleman[1] · J. D. Mitchell[1] · F. L. Smith[1] · M. Tsalakou[1]

Received: 9 May 2023 / Accepted: 4 April 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.

Keywords Semigroup · Monoid · Finite presentations · Congruences · Algorithms · Computer algebra

Contents

Communicated by Markus Lohrey.

✉ J. D. Mitchell
jdm3@st-andrews.ac.uk

T. D. H. Coleman
tdhc@standrews.ac.uk

F. L. Smith
fls3@st-andrews.ac.uk

M. Tsalakou
mt200@st-andrews.ac.uk

1 School of Mathematics and Statistics, University of St Andrews, St Andrews, UK

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 Sect. 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 $\langle a, b \mid ab^n, a^n b^{n+1}, b^n ab^{-1}a^{-1}\rangle$ 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 Sect. 6; see also Sect. 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 Sect. 2; in Sect. 3 we define the infinite family of procedures for enumerating a right congruence on a monoid; the validity of congruence enumeration is shown in Sect. 4; in Sect. 5 we discuss how congruence enumeration can be used to compute congruences of a monoid that is not given *a priori* by a finite presentation; in Sects. 6 and 7 we describe the two main strategies for congruence enumeration; in Sect. 8 we discuss some issues related to the implementation of congruence enumeration; and, finally, in Sect. 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 μ is an equivalence relation on a set X, then we denote by x/μ the equivalence class of x in μ and by X/μ the set $\{ x/\mu \ : \ x \in X \}$ of equivalence classes. The least equivalence relation on X with respect to containment is $\Delta_X = \{ (x, x) \in X \times X \ : \ x \in X \}$; 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 $s_i = p_i u_i q_i$, $s_{i+1} = p_i v_i q_i$, 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 $\Psi : X \times S \longrightarrow X$ is a *right action* of S on X if $((x, s)\Psi, t)\Psi = (x, st)\Psi$ 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 $\Psi : X \times S \longrightarrow X$ is a right action of a semigroup S on a set X, then $\Psi^1 : X \times S^1 \longrightarrow X$ 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 $f : X \longrightarrow Y$, 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 Ψ of a semigroup S on a set X induces a homomorphism $\Psi' : S \longrightarrow X^X$ defined by $(s)\Psi' : x \mapsto x \cdot s$ for all $s \in S$ and all $x \in X$.

The *kernel* of a right action Ψ of a semigroup S on a set X is the kernel of the function $\Psi' : S \longrightarrow X^X$,

$$\begin{aligned}\ker(\Psi) &= \{ (s, t) \ : \ (s)\Psi' = (t)\Psi' \} \\ &= \{ (s, t) \in S \times S \ : \ (x, s)\Psi = (x, t)\Psi, \text{ for all } x \in X \}.\end{aligned}$$

[1]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.

[1] This 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 Ψ is an action, we will never use $\ker(\Psi)$ to mean its kernel as a function.

$$\begin{tikzcd}
X \times S \arrow[r] \arrow[d] & Y \times S \arrow[d] \\
X \arrow[r, "\lambda"] & Y
\end{tikzcd}$$

Fig. 1 A commutative diagram illustrating a homomorphism of actions where $X \times S \longrightarrow Y \times S$ is the function defined by $(x, s) \mapsto ((x)\lambda, s)$

If S acts on the sets X and Y on the right, then we say that $\lambda : X \longrightarrow Y$ 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. 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 $(a_1, \dots, a_n)$ where $a_1, \dots, a_n \in A$ and $n \geq 0$; say that the *length* of the word is n. We will write $a_1 \cdots a_n$ rather than $(a_1, \dots, a_n)$, 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 ε 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 ρ be a congruence of S, and let σ be a right congruence of S such that $\rho \subseteq \sigma$. The set σ/ρ defined as

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

is a right congruence on S/ρ.

Proposition 2.1 *Let S be a semigroup, let ρ be a congruence of S, and let σ be a right congruence of S such that $\rho \subseteq \sigma$. The right actions of S on $(S/\rho)/(\sigma/\rho)$ and S/σ defined by*

$$(u/\rho)/(\sigma/\rho) \cdot v = (uv/\rho)/(\sigma/\rho) \quad \textit{and} \quad u/\sigma \cdot v = uv/\sigma \quad \textit{for all} \quad u, v \in S$$

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 $\Psi : X \times S \longrightarrow X$ be a right action. If $\mu \subseteq \ker(\Psi)$ is a congruence on S, then $\overline{\Psi} : X \times S/\mu \longrightarrow X$ 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 Ψ.

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 Sects. 6 and 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 $\mathcal{P} = \langle A \mid R \rangle$ 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 ρ the least right congruence on A^* containing $R^{\#}$ and S. In this notation, the monoid M defined by the presentation $\mathcal{P}$ is $A^*/R^{\#}$. If the number of congruence classes of ρ is finite, then the output of a congruence enumeration is a description of the natural right action of A^* on the congruence classes of ρ. 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 $(A^*/\rho)^{(A^*/\rho)}$ on the set of congruence classes of ρ. 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 $\langle A \mid R \rangle$ to terminate cannot be computed as a function of $|A|$ and $|R|$. If the number of congruence classes of ρ 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 $\mathcal{P}$ 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 ρ 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 $\{\, (u^\dagger, v^\dagger) \;:\; (u, v) \in W \,\}$.

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 ρ 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 Sect. 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 Γ as a *word graph*. The word graph $\Gamma = (\{0\}, \varnothing)$ is the *trivial word graph*.

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

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

If $\alpha, \beta \in N$, then an (α, β)-*path* is a sequence of edges $(\alpha_1, a_1, \alpha_2), \ldots, (\alpha_n, a_n, \alpha_{n+1}) \in E$ where $\alpha_1 = \alpha$ and $\alpha_{n+1} = \beta$ and $a_1, \ldots, a_n \in A$; α is the *source* of the path; the word $a_1\cdots a_n \in A^*$ *labels* the path; β is the *target* of the path; and the *length* of the path is n. If $\alpha, \beta \in N$ and there is an (α, β)-path in Γ, then we say that β is *reachable* from α. 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* Γ is the function $\pi_\Gamma : N \longrightarrow \mathfrak{P}(A^* \times A^*)$ defined by $(u, v) \in (\alpha)\pi_\Gamma$ if there exists a node β such that u and v both label (α, β)-paths in Γ. If Γ is a word graph and α is a node in Γ, then $(\alpha)\pi_\Gamma$ is reflexive and symmetric, and $(\alpha)\pi_\Gamma$ is transitive for all α if and only if Γ is deterministic. In particular, $(\alpha)\pi_\Gamma$ is an equivalence relation for all α if and only if Γ is deterministic. If $R \subseteq A^* \times A^*$, Γ is a word graph, and π_Γ is the path relation of Γ, then we say that Γ is *compatible with* R if $R \subseteq (\alpha)\pi_\Gamma$ for every node α in Γ.

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* Γ/κ *of* Γ *by* κ 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, κ is any equivalence relation on N, and κ' is any equivalence relation on the set of nodes of Γ/κ, then $(\Gamma/\kappa)/\kappa' = \Gamma/\kappa''$ where κ'' is the least equivalence on N containing

κ 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 $\phi : N_1 \longrightarrow N_2$ is a *homomorphism* if $(\alpha, a, \beta) \in E_1$ implies $((\alpha)\phi, a, (\beta)\phi) \in E_2$; and we write $\phi : \Gamma_1 \longrightarrow \Gamma_2$. An *isomorphism* of word graphs Γ_1 and Γ_2 is a bijection $\phi : \Gamma_1 \longrightarrow \Gamma_2$ such that both ϕ and ϕ^{-1} are homomorphisms. If κ is any equivalence relation on a word graph $\Gamma = (N, E)$, then the function $\phi : \Gamma \longrightarrow \Gamma/\kappa$ defined by $(\alpha)\phi = \alpha/\kappa$ is a homomorphism. Conversely, if $\phi : \Gamma_1 \longrightarrow \Gamma_2$ is a homomorphism, then $(\Gamma_1)\phi := \{\ (\alpha)\phi \in N_2\ :\ \alpha \in N_1\ \}$ is isomorphic to the quotient $\Gamma_1/\ker(\phi)$. If $(\alpha_1, a_1, \alpha_2), \ldots, (\alpha_n, a_n, \alpha_{n+1}) \in E_1$ is a path in Γ_1 and $\phi : \Gamma_1 \longrightarrow \Gamma_2$ is a homomorphism, then, by definition, $((\alpha_1)\phi, a_1, (\alpha_2)\phi), \ldots, ((\alpha_n)\phi, a_n, (\alpha_{n+1})\phi) \in E_2$ is a path in Γ_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 $\pi_{\Gamma_1} : N_1 \longrightarrow \mathfrak{P}(A^* \times A^*)$ and $\pi_{\Gamma_2} : N_2 \longrightarrow \mathfrak{P}(A^* \times A^*)$ be the path relations of Γ_1 and Γ_2, respectively. If $\phi : \Gamma_1 \longrightarrow \Gamma_2$ 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 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 $f : C \longrightarrow N$ 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 Γ 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 $\langle A | R \rangle$ 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 $i \in \mathbb{N}$) is (Γ_i, κ_i) for some word graph Γ_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 α is a node in Γ_i and there is no edge in Γ_i labelled by $a \in A$ with source α, then we define Γ_{i+1} to be the word graph obtained from Γ_i by adding the new node $\beta := 1 + \max \bigcup_{j \leq i} N_j$ and the edge (α, a, β). We define $\kappa_{i+1} := \kappa_i \cup \{(\beta, \beta)\}$.

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

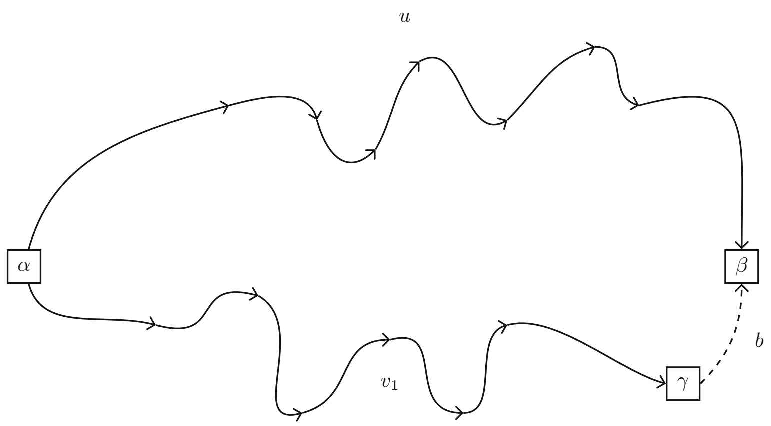


Fig. 2 A diagram representing **TC2**(a), solid lines correspond to paths in Γ_i and the dashed edge is the one defined in **TC2**(a)

(a) If u and v_1 label paths from α to some nodes $\beta, \gamma \in N_i$ in Γ_i, respectively, but γ is not the source of any edge labelled by b, then we set Γ_{i+1} to be the word graph obtained from Γ_i by adding the edge (γ, b, β) and we define $\kappa_{i+1} := \kappa_i$ (Fig. 2).

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

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

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 Γ_{i+1} to be the quotient of Γ_i by κ_i and define κ_{i+1} to be the least equivalence on N_{i+1} containing every (β, γ), $\beta \neq \gamma$, for which there exist $\alpha \in N_{i+1}$ and $a \in A$ such that $(\alpha, a, \beta), (\alpha, a, \gamma) \in E_{i+1}$. Recall that the quotient Γ_i/κ_i of Γ_i by κ_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 Γ_{i+1} is set to be equal to the minimum of the set of nodes in its equivalence class in κ_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 κ_{i+1} will equal Δ_{N_i}, and Γ_{i+1} and Γ_i will be equal.

If w labels an (α, β)-path P in some Γ_i, then neither **TC1** nor **TC2** changes any of the edges belonging to P. Hence if Γ_{i+1} is obtained from Γ_i by applying **TC1** or **TC2**, then w labels an (α, β)-path in Γ_{i+1} also. If Γ_{i+1} is obtained from Γ_i by applying **TC3**, then Γ_{i+1} is a homomorphic image of Γ_i. As already noted, homomorphisms preserve paths, and so w labels a path in Γ_{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 ρ 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 $\pi_{\Gamma_1} : N_1 \longrightarrow \mathfrak{P}(A^* \times A^*)$ such that $(0)\pi_{\Gamma_1} \subseteq \rho$ and $\kappa_1 = \Delta_{N_1}$. Then a *congruence enumeration for* ρ with input (Γ_1, κ_1) consists of:

(a) For every $(u, v) \in S$, by repeatedly applying **TC1** (if necessary), add edges to Γ_1 so that it contains paths labelled by u and v both with source 0.
(b) Apply **TC2** to 0 and every $(u, v) \in S$.

If (Γ_m, κ_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 α with label $a \in A$, then there exists $j \geq i$ such that either: α is no longer a node in Γ_j; or there is an edge incident to α with label a in Γ_j.
(d) If $\alpha \in N_i$ and $(u, v) \in R$, then there exists $j \geq i$ such that either: α is no longer a node in Γ_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 Γ_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 $\langle A|R\rangle$; see Sect. 5 for more details.

A congruence enumeration *terminates* if the output (Γ_i, κ_i) has the property that applying any of **TC1**, **TC2**, or **TC3** to (Γ_i, κ_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 Γ_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 Sects. 6 and 7.

4 Validity

In this section we address the validity of congruence enumeration as defined in Sect. 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 ρ is the least right congruence on A^* containing $R^\#$ and S, then the following hold:*

(a) *If a congruence enumeration for ρ terminates with output word graph $\Gamma = (N, E)$, then A^*/ρ is finite and the function $\phi : N \times A^* \longrightarrow N$, defined by $(\alpha, w)\phi = \beta$*

whenever w labels an (α, β)-path in Γ, is a right action that is isomorphic to the natural action of A^* on A^*/ρ by right multiplication.

(b) *If A^*/ρ is finite, then any congruence enumeration for ρ 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 $\phi : N \times A^* \longrightarrow N$ defined by $(\alpha, w)\phi = \beta$ whenever w labels an (α, β)-path in Γ 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 Sect. 4.2. We start by showing that the steps **TC1**, **TC2**, and **TC3** preserve certain properties of word graphs in Proposition 4.4. In Sect. 4.1, we show that every congruence enumeration eventually stabilises and is eventually compatible with R; and in Sect. 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 (Γ_i, κ_i) where every node in Γ_i is reachable from 0, then every node in the output Γ_{i+1} 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 (Γ_i, κ_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 $(0)\pi_{\Gamma_i} \subseteq (0)\pi_{\Gamma_{i+1}} \subseteq \rho$, and $(0)\pi_{\Gamma_i/\kappa_i} \subseteq (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}} \subseteq \rho$.*

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 a. In this case, Γ_{i+1} is obtained from Γ_i by adding the single node $\beta := 1 + \min \bigcup_{j \leq i} N_j$ and the single edge (α, a, β), and $\kappa_{i+1} := \kappa_i \cup \{(\beta, \beta)\}$. In **TC2**(a), there exists $(u, v) \in R$, $b \in A$, and $v_1 \in A^*$ 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 $\kappa_{i+1} := \kappa_i$.

Hence if either **TC1** or **TC2**(a) is applied, then the identity map $N_i \longrightarrow N_{i+1}$ is a homomorphism from Γ_i to Γ_{i+1} and so the identity map $N_i/\kappa_i \longrightarrow N_{i+1}/\kappa_i$ is also a homomorphism from Γ_i/κ_i to $\Gamma_{i+1}/\kappa_{i+1}$. Hence, by Lemma 3.2, $(0)\pi_{\Gamma_i} \subseteq (0)\pi_{\Gamma_{i+1}}$ and $(0)\pi_{\Gamma_i/\kappa_i} \subseteq (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}}$.

To show that $(0)\phi_{\Gamma_{i+1}} \subseteq \rho$ we consider the cases when **TC1** and **TC2(a)** are applied separately.

TC1: Suppose that $(u, v) \in (0)\pi_{\Gamma_{i+1}} \backslash (0)\pi_{\Gamma_i}$ 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, a, β) must occur at least once in both paths. But β is the source of no edges in Γ_{i+1}, and so (α, a, β) occurs once, and it must be the last edge, in both paths. Hence $u = u_1 a$ and $v = v_1 a$ where $u_1, v_1 \in A^*$ label $(0, \alpha)$-paths in Γ_i. Therefore $(u_1, v_1) \in \rho$ by assumption and since ρ is a right congruence, $(u, v) = (u_1 a, v_1 a) \in \rho$ as well. Hence $(0)\pi_{\Gamma_{i+1}} \subseteq \rho$.

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 $(x, y) \in (0)\pi_{\Gamma_{i+1}}$, 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 and Y is strictly less than k, then $(x, y) \in \rho$. The base case is when $k = 0$ and, in this case, $(x, y) \in (0)\pi_{\Gamma_i} \subseteq \rho$, as required.

Suppose that $k \geq 1$, that $x, y \in A^*$ are any words labelling $(0, \delta)$-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 $e_1, \ldots, e_r \in E_{i+1}$, and $j \in \{1, \ldots, r\}$ is the largest value such that $e_j = (\gamma, b, \beta)$, then we may write $x = x_1 b x_2$ where x_1 labels the $(0, \gamma)$-path $(e_1, \ldots, e_{j-1})$ and x_2 labels the (β, δ)-path $(e_{j+1}, \ldots, e_r)$. If w' labels any path from 0 to α in Γ_i, then $w' v_1$ labels an $(0, \gamma)$-path in Γ_i. It follows that $(x_1, w' v_1) \in (0)\pi_{\Gamma_{i+1}}$. The number of occurrences of (γ, b, β) in $(e_1, \ldots, e_{j-1})$ is strictly less than k, and the number in the path (in Γ_i) labelled by $w' v_1$ is 0. Hence, by the inductive hypothesis, $(x_1, w' v_1) \in \rho$ and so $(x, w' v x_2) = (x_1 b x_2, w v_1 b x_2) \in \rho$.

The word w' labels a $(0, \alpha)$-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 $w' u x_2$ labels a $(0, \delta)$-path in Γ_i, which therefore contains no occurrences of the edge (γ, b, β). The word y also labels the $(0, \delta)$-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, $(w' u x_2, y) \in \rho$. Finally, since $(u, v) \in R \subseteq \rho$, $(w' u x_2, w' v x_2) \in R^{\#} \subseteq \rho$ also. Hence $(x, w' v x_2), (w' v x_2, w' u x_2), (w' u x_2, y) \in \rho$ and so by transitivity, $(x, y) \in \rho$.

We conclude the proof by showing that $(0)\pi_{\Gamma_{i+1}/\kappa_{i+1}} \subseteq \rho$ when either **TC1** or **TC2**(a) is applied. Suppose that $(u, v) \in (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}}$. If $(u, v) \in (0)\pi_{\Gamma_{i+1}}$, then $(u, v) \in \rho$, as required. If $(u, v) \notin (0)\pi_{\Gamma_{i+1}}$, then u and v label $(0, \zeta)$- and $(0, \eta)$-paths in Γ_{i+1}, respectively, for some nodes $\zeta \neq \eta$. Recall that $\kappa_{i+1} = \kappa_i \cup \{(\beta, \beta)\}$ if **TC1** is applied, and $\kappa_{i+1} = \kappa_i$ if **TC2**(a) is applied. In either case, $(\zeta, \eta) \in \kappa_{i+1}$ and $\zeta \neq \eta$ implies that $(\zeta, \eta) \in \kappa_i$. 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 $u', v' \in A^*$ labelling $(0, \zeta)$- and $(0, \eta)$-paths in Γ_i, respectively. Hence $(u, u'), (v', v) \in (0)\pi_{\Gamma_{i+1}} \subseteq \rho$ and, by construction, $(u', v') \in (0)\pi_{\Gamma_i/\kappa_i} \subseteq \rho$. Thus, by transitivity, $(u, v) \in \rho$, 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 $(u, v) \in R$ 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, $\Gamma_{i+1} = \Gamma_i$ and κ_{i+1} is the least equivalence relation containing κ_i and (β, γ). If **TC3** is applied, then $\Gamma_{i+1} = \Gamma_i/\kappa_i$ and κ_{i+1} is the least equivalence on N_{i+1} containing every (δ, ζ), $\delta \neq \zeta$, for which there exist $\eta \in N_{i+1}$ and $a \in A$ such that $(\eta, a, \delta), (\eta, a, \zeta) \in E_{i+1}$.

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

Hence, in both cases, it suffices to show that $(0)\pi_{\Gamma_i/\kappa_i} \subseteq (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}} \subseteq \rho$. We consider the cases when **TC2**(c) and **TC3** are applied separately.

TC2(c): If β and γ are the nodes defined above, and σ is the least equivalence relation on the set of nodes in Γ_i/κ_i containing β/κ_i and γ/κ_i, then $(\Gamma_i/\kappa_i)/\sigma = \Gamma_i/\kappa_{i+1} = \Gamma_{i+1}/\kappa_{i+1}$. Hence $\Gamma_{i+1}/\kappa_{i+1}$ is a homomorphic image of Γ_i/κ_i and so $(0)\pi_{\Gamma_i/\kappa_i} \subseteq (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}}$.

Suppose that $(x, y) \in (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}}$. Then x and y both label $(0, \delta)$-paths in $\Gamma_{i+1}/\kappa_{i+1}$ for some node δ in $\Gamma_{i+1}/\kappa_{i+1}$. It follows that there are nodes ζ and η in Γ_{i+1} such that $(\delta, \zeta), (\delta, \eta) \in \kappa_{i+1}$ and x and y label $(0, \zeta)$- and $(0, \eta)$-paths in $\Gamma_{i+1} = \Gamma_i$, respectively.

If $(\zeta, \eta) \in \kappa_i$, then $(u, v) \in (0)\pi_{\Gamma_i/\kappa_i}$ and so $(u, v) \in \rho$ by assumption. If $(\zeta, \eta) \notin \kappa_i$, then without loss of generality we may assume that $(\zeta, \eta) = (\beta, \gamma)$. In this case, if $w \in A^*$ labels a $(0, \alpha)$ path in Γ_i, then wu and x both label $(0, \beta) = (0, \zeta)$-paths in Γ_i, and so $(wu, x) \in (0)\pi_{\Gamma_i} \subseteq \rho$. Similarly, wv and y both label $(0, \gamma)$-paths in Γ_i, and so $(wv, y) \in \rho$, also. Finally, $(u, v) \in R$ and so $(wu, wv) \in R^{\#} \subseteq \rho$, and so $(x, y) \in \rho$ also.

TC3: Clearly, $\Gamma_{i+1}/\kappa_{i+1}$ is a homomorphic image of Γ_{i+1} and Γ_{i+1} is defined to be Γ_i/κ_i in **TC3**. Hence $(0)\pi_{\Gamma_i/\kappa_i} \subseteq (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}}$ by Lemma 3.2. It remains to show that $(0)\pi_{\Gamma_{i+1}/\kappa_{i+1}} \subseteq \rho$.

Suppose that $(x, y) \in (0)\pi_{\Gamma_{i+1}/\kappa_{i+1}}$. Then there exist nodes ζ and η in Γ_{i+1} such that $(\zeta, \eta) \in \kappa_{i+1}$ where x labels a $(0, \zeta)$-path and y labels a $(0, \eta)$-paths in Γ_{i+1}. In **TC3**, κ_{i+1} is defined to be the least equivalence containing $(\beta, \gamma) \in N_{i+1} \times N_{i+1}$ such that there are edges (α, a, β) and (α, a, γ) in Γ_{i+1}. Hence $(\eta, \zeta) \in \kappa_{j+1}$ implies that there exist $\alpha_1, \ldots, \alpha_{n-1}, \beta_1 := \zeta, \beta_1, \ldots, \beta_n := \eta \in N_{j+1}$ and $a_1, \ldots, a_{n-1} \in A$ where $(\alpha_j, a_j, \beta_j), (\alpha_j, a_j, \beta_{j+1}) \in E_{j+1}$ and $\beta_j \neq \beta_{j+1}$ for every j. Suppose that $w_j \in A^*$ is any word labelling a $(0, \alpha_j)$-path for $j = 1, \ldots, n-1$, that $x_1 = x$, that x_j labels a $(0, \beta_j)$-path for $j = 2, \ldots, n-1$, and that $x_n = y$.

Then $w_j a_j$ and x_j both label $(0, \beta_j)$-paths in Γ_{i+1} for every j. Similarly, $w_j a_j$ and x_{j+1} both label $(0, \beta_{j+1})$-paths in Γ_{i+1} for every j. Hence $(w_j a_j, x_j), (w_i a_j, x_{j+1}) \in (0)\pi_{\Gamma_{i+1}} \subseteq \rho$ for every j. Therefore $(x_j, x_{j+1}) \in \rho$ for every j, and so by transitivity $(x_1, x_n) = (x, y) \in \rho$, as required. □

4.1 Completeness, determinism, and compatibility

In this section, we will prove that if at some step i in a congruence enumeration the word graph Γ_i is complete, then that enumeration terminates, and that every congruence enumeration is eventually compatible with ρ.

We say that a congruence enumeration *stabilises* if for every $i \in \mathbb{N}$ and every node α of Γ_i there exists $K \in \mathbb{N}$ such that for all $j \geq K$ either α is not a node in Γ_j or if $(\alpha, a, \beta) \in E_j$, then $(\alpha, a, \beta) \in E_{j+1}$ for any $a \in A$.

Lemma 4.7 *Every congruence enumeration stabilises.*

Proof We will prove a slightly stronger statement, that any sequence of **TC1**, **TC2**, and **TC3** stabilises.

Suppose that $i \in \mathbb{N}$, α is an arbitrary node in Γ_i, and $a \in A$ is arbitrary. Either there exists $K > i$ such that α is no longer a node in Γ_K, or α is a node in Γ_j for all $j \geq i$. If α is no longer a node in Γ_K, then, from the definitions of **TC1**, **TC2**, and **TC3**, α is not a node in Γ_k for any $k \geq K$.

Suppose that α is a node in Γ_j for all $j \geq i$ and that $(\alpha, a, \beta_1) \in E_j$ for some $\beta_1 \in A^*$. If $\beta_1 \in N_j$ is replaced by $\beta_2 \in N_k$ at some step $k \geq j$, then step k is an application of **TC3**. In particular, $0 \leq \beta_2 \leq \beta_1$. This process can be repeated only finitely many times because there are only finitely many natural numbers less than β_1. □

If Γ_i is complete for some i, then **TC1** cannot be applied again at any step after i. In this case, **TC2** and **TC3** make κ coarser and hence reduce the number of nodes in Γ_i. Since the number of nodes in Γ_i is finite, it follows that the enumeration must terminate at some point. We record this in the following corollary.

Corollary 4.8 *If Γ_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 Γ_i starting at 0 for some $i \in \mathbb{N}$, then there exists $j \geq i$ such that w labels a unique path in Γ_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 Γ_i starting (and ending) at 0.

Suppose that every word of length at most $n - 1$ labelling a path starting at 0 in Γ_i labels a unique path, and let $w \in A^*$ be any word of length $n \geq 1$, labelling a path in Γ_i starting at 0. If $w = w_1 a$ for some $w_1 \in A^*$ and $a \in A$, then w_1 labels a unique path in Γ_i starting at 0 by induction. Suppose that α is the target of the unique path labelled w_1. By Lemma 4.7, we may assume without loss of generality that α is a node in Γ_j for every $j \geq i$. If there exist edges $(\alpha, a, \beta_1), \ldots, (\alpha, a, \beta_r)$ in Γ_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 (e) of the definition of a congruence enumeration (Definition 3.3), there exists $j > i$ such that $\kappa_j = \Delta_{N_j}$ and so at step j, there is only one edge incident to α labelled a. □

Next we show that every congruence enumeration is eventually compatible with ρ.

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 $(w_j, w_{j+1}) \in (0)\pi_{\Gamma_i}$ 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 Γ_j for all $j \geq i$ for some sufficiently large i. We may choose i large enough so that the target node α of this path is a node in every Γ_j for $j \geq i$. By Definition 3.3(d), for sufficiently large i, $(x, y) \in (\alpha)\pi_{\Gamma_i}$ and so $(px, py) \in (0)\pi_{\Gamma_i}$. 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 $(w_j, w_{j+1}) = (pxq, pyq) \in (0)\pi_{\Gamma_i}$. □

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 ρ 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 (Γ_m, κ_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), **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 Γ_i, and so $(xq, yq) \in (0)\pi_{\Gamma_i}$. □

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, Γ_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 $(u, v) \in \rho$. □

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* α *in* Γ_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) ⇒ (ii).** If $(u, v) \in R^{\#}$ and $w \in A^*$ is arbitrary, then $(wu, wv) \in R^{\#}$ and so, by Lemma 4.10, $(wu, wv) \in (0)\pi_{\Gamma_i}$ for sufficiently large i. In particular, for large enough i, wu and wv both label $(0, \beta)$-paths for some node β. If α is the target of the path starting at 0 labelled by w, then u and v both label a (α, β)-paths in Γ_i and so $(u, v) \in (\alpha)\pi_{\Gamma_i}$ for i large enough.
(ii) ⇒ (iii) This follows immediately since 0 is a node in every word graph.
(iii) ⇒ (i) Follows by applying Corollary 4.12 to $\rho = R^{\#}$. □

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* ρ *is the least right congruence on* A^* *containing* $R^{\#}$ *and* S, *then the following hold:*

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

Proof We start the proof by showing that there is always a bijection between a certain set of nodes in the word graphs Γ_i (defined in a moment) and the classes of ρ. 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_α labels a path from 0 to α in every Γ_i for sufficiently large i. We define $f : M \longrightarrow A^*/\rho$ by

$$(\alpha)f = w_\alpha/\rho. \tag{4.1}$$

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_α and w_β both label paths from 0 to the same node in every Γ_i for large enough i. By Lemma 4.7, we can also choose i large enough so that the target γ of these paths is a node in Γ_j for all $j \geq i$. But w_α and w_β also label $(0, \alpha)$- and $(0, \beta)$-paths in every Γ_i for sufficiently large i. By Lemma 4.9, there exists $j \geq i$ such that w_α labels a unique path starting at 0, and so $\alpha = \gamma$. Similarly, there exists $k \geq j$ such that w_β 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 Γ_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 Γ_j for $j \geq i$. In particular, u

Fig. 3 The commutative diagram in the proof of Theorem 4.1(a)

$$\begin{tikzcd}
N \times A^* \arrow[r, "g"] \arrow[d, "\phi"] & A^*/\rho \times A^* \arrow[d, "\Phi"] \\
N \arrow[r, "f"] & A^*/\rho
\end{tikzcd}$$

labels a $(0, \alpha)$-path in Γ_i for some $\alpha \in M$. But w_α also labels a $(0, \alpha)$-path in Γ_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 (Γ, κ) is the output of the enumeration where $\Gamma = (N, E)$ and that $\pi : N \longrightarrow \mathfrak{P}(A^* \times A^*)$ is the path relation of Γ. Since the congruence enumeration process has terminated, Γ is finite and its set of nodes N coincides with the set M defined at the start of the proof. Hence A^*/ρ is finite, since $f : N \longrightarrow A^*/\rho$ defined in (4.1) is a bijection.

Since any application of **TC3** results in no changes to Γ, it follows that $\kappa = \Delta_N$. Hence Γ is deterministic, and so every $w \in A^*$ labels exactly one path starting at every $u \in N$. In particular, ϕ (as defined in the statement) is a well-defined function. If $w_1, w_2 \in A^*$ label (α_1, α_2)- and (α_2, α_3)-paths in Γ, respectively, then certainly w_1w_2 labels a (α_1, α_3)-path, and so ϕ is an action.

We will show that $f : N \longrightarrow A^*/\rho$ defined in (4.1) is an isomorphism of the actions ϕ and the natural action $\Phi : A^*/\rho \times A^* \longrightarrow A^*/\rho$ defined by $(u/\rho, v)\Phi = uv/\rho$. We define $g : N \times A^* \longrightarrow (A^*/\rho) \times A^*$ 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. 3 commutes. Suppose that $(\alpha, v) \in N \times A^*$. Then

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

and if v labels a (α, β)-path in Γ, then

$$((\alpha, v)\phi)f = (\beta)f = w_\beta/\rho.$$

But w_β and $w_\alpha v$ both label $(0, \beta)$-paths in Γ 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 $((\alpha, v)g)\Phi = ((\alpha, v)\phi)f$ and so f is homomorphism of the actions ϕ and Φ.

(b). If A^*/ρ 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 Γ_i or κ_i, $i \geq K$, and so the enumeration terminates. $\square$

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 $R^\#$ 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 $\phi : N \times A^* \longrightarrow N$ defined by*

Table 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. 4

0	1	2	3	4	5	6	7	8
ε	a	b	c	a^2	ab	ba	bc	bab

$(\alpha, w)\phi = \beta$ whenever w labels an (α, β)-path in Γ 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 $S = \varnothing$ in Theorem 4.1, then the least right congruence ρ containing S and $R^\#$ is just $R^\#$. By Theorem 4.1, the action $\phi : N \times A^* \longrightarrow N$ is isomorphic to the natural action of A^* on $A^*/R^\#$ by right multiplication. By Corollary 4.13, the kernel of ϕ 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 ϕ 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 $\phi : A^* \longrightarrow M$ 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 Γ with nodes $N = \{0, \ldots, |M| - 1\}$ and edges (α, a, β) for all $\alpha, \beta \in N$ and every $a \in A$ such that $m_\alpha a = m_\beta$. The right Cayley graph Γ is complete and deterministic, and so if $\pi : N \longrightarrow \mathfrak{P}(A^* \times A^*)$ is the path relation on Γ, 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

$$0 + 1 = 1 + 0 = 1 + 1 = 1, \qquad 0 + 0 = 0$$

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. 4; see also Table 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

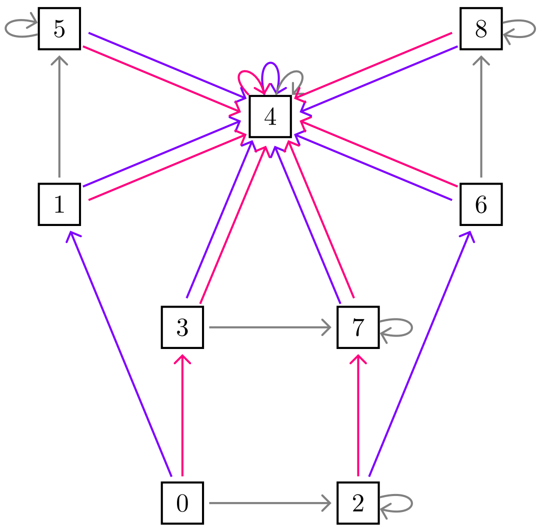


Fig. 4 The right Cayley graph of the monoid M from Example 5.1 with respect to the generating set $\{a, b, c\}$; see Table 1 for a representative word for each node. Purple arrows correspond to edges labelled a, gray labelled b, and pink labelled c

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, $(0)\pi_{\Gamma_1} = R^\#$ and so (Γ_1, Δ_{N_1}) is a valid input to a congruence enumeration. The right Cayley graph Γ_1 is already complete, deterministic, and compatible with $R^\#$ by definition. It follows that **TC2** and **TC3** are the only steps actually applied in a congruence enumeration with input Γ_1. More precisely, **TC2**(c) is applied to 0 and every $(u, v) \in S$, and then **TC3** 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 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 ρ on M containing $S = \{(a, b)\}$ we perform the following steps.

Set Γ_1 to be the right Cayley graph of M with respect to the generating set $\{a, b, c\}$; see Fig. 4 and Table 1. Suppose that $a < b < c$.

Step 1 Apply **TC2** to the only pair (a, b) in S, and the output of this is (Γ_2, κ_2) where κ_2 is the least equivalence on the nodes of Γ_2 containing $(1, 2)$; see Fig. 5a.

Step 2 Apply **TC3** to obtain the quotient of Γ_2 by κ_2, the output is (Γ_3, κ_3) where $\Gamma_3 = \Gamma_2/\kappa_2$ and κ_3 is the least equivalence containing $(1, 5)$, $(4, 6)$, and $(4, 7)$; see Fig. 5b.

Step 3 Apply **TC3** with input (Γ_3, κ_3) and output (Γ_4, κ_4) where $\Gamma_4 = \Gamma_3/\kappa_3$ and κ_4 is the least equivalence containing $(4, 8)$; see Fig. 5c.

Step 4 Apply **TC3** with input (Γ_4, κ_4) and output (Γ_5, κ_5) where $\Gamma_5 = \Gamma_4/\kappa_4$ and $\kappa_5 = \Delta_{N_5}$. Since Γ_5 is complete and compatible with ρ and κ_5 is trivial, the enumeration terminates with Γ_5; see Fig. 5d.

We conclude that ρ has 4 equivalence classes whose representatives ε, a, c, and a^2 correspond to the nodes $\{0, 1, 3, 4\}$ in the graph Γ_5. By following the paths in Γ_5 starting at 0 labelled by each of the words in Table 1 we determine that the congruence classes of ρ are: $\{\varepsilon\}$, $\{a, b, ab\}$, $\{c\}$, $\{a^2, ba, bc, bab\}$.

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 Γ_i such that there exists $(u, v) \in R$ 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_1 label paths starting at α where $v = v_1 b$ for some $b \in A$, and then apply **TC2** where part (a) applies. In the latter case, we simply apply **TC2** to α and $(u, v) \in R$ where part (c) applies.

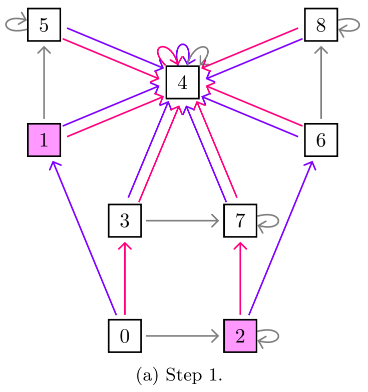
HLT2. Repeatedly apply **TC3** until the resulting κ_{i+1} is trivial.

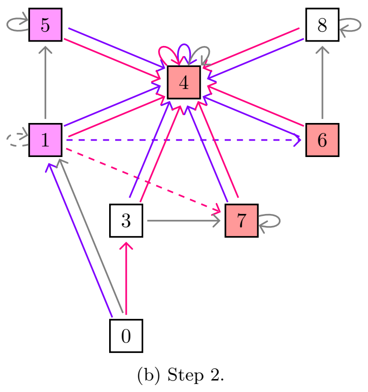
HLT3. If α is the node from **HLT1**, then we apply **TC1** to α and every $a \in A$, if any, such that there is no edge incident to α labelled by a.

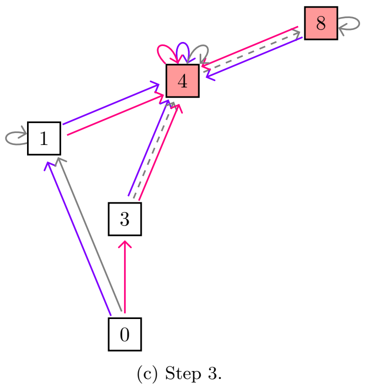
We note that if for every $a \in A$ there exists $(u, v) \in R$ such that either u or v starts with a, then **HLT3** does not have to be performed.

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

Proposition 6.1 *If $\langle A|R\rangle$ 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 ρ is the least right congruence on A^* containing S and $R^\#$, then the HLT strategy applied to* (Γ_1, κ_1)







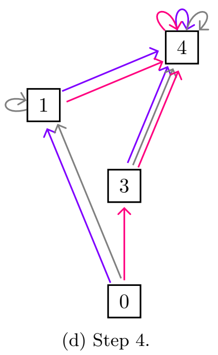


Fig. 5 The output (Γ_i, κ_i) of each step in Example 5.2. Purple arrows correspond to a, gray to b, pink to c, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes; see Table 1 for a representative word for each node. A dashed edge with a single arrowhead denotes an edge that is obtained from **TC2** or **TC3**, solid edges correspond to edges that existed at the previous step

where $\Gamma_1 = (N_1, E_1)$ *is a word graph such that* $(0)\pi_{\Gamma_1} \subseteq \rho$ *and* $\kappa_1 = \Delta_{N_1}$ *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 $\Gamma_i = (N_i, E_i)$ such that there is no edge labelled by some $a \in A$ incident to α. If $\alpha \in N_j$ for all $j \geq i$, then there exists $k \in \mathbb{N}$ such that α is the minimum node to which **HLT1** is applied. If there exists $(u, v) \in R$ 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 Γ_i and $(u, v) \in R$ are such that u and v both label paths starting at α. If $\alpha \in N_j$ for all $j \geq i$, then there exists $k \in \mathbb{N}$ 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 $(\beta, \gamma) \in \kappa_i$. But **TC3** is applied at some later step, and so β and γ are identified in the corresponding quotient. Eventually

Table 2 A word labelling a path from 0 to each node in the right Cayley graph on the monoid M from Example 6.2; see Fig. 6

0	1	2	3	4
ε	a	b	a^2	c

the paths labelled by u and v starting at α end at the same node, i.e. $(u, v) \in (\alpha)\pi_{\Gamma_i}$ for sufficiently large i.
(e). If $(\beta, \gamma) \in \kappa_i$ for some i, then there exists $j \geq i$ such that **TC3** is applied by the definition of **HLT2**. If we choose the smallest such j, then $\kappa_i \subseteq \kappa_j$ and so $(\beta, \gamma) \notin \kappa_{j+1}$. □

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

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

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 Γ_1 is the trivial graph and the input κ_1 is Δ_{N_1}. We suppose that $a < b < c$.

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 κ_2 to be the least equivalence containing $\kappa_1 = \Delta_{N_1}$ and $(1, 2)$. The output is graph Γ_2 in Fig. 6a.

Step 2 We continue with Definition 3.3(b). We apply **TC3** and we get the graph Γ_3 in Fig. 6b which is the quotient of Γ_2 by κ_2. The set κ_3 is defined to be Δ_{N_3}.

Step 3 Now we are ready to apply **HLT1**. Since 0 is the minimum node in Γ_3 such that $(ac, a^2) \in R$ and ac and a^2 do not label paths in Γ_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 Γ_4 in Fig. 6c.

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 (Γ_i, κ_i), $i = 5, \ldots, 9$ are shown in Fig. 6d–h.

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 Γ_9. We apply **HLT1** to the node 1 and the relation $(ac, a^2) \in R$ and the output is graph Γ_{10} in Fig. 6i.

Step 10 We apply **HLT1** to the node 1 and the relation $(cb, bc) \in R$ and the output is graph Γ_{11} in Fig. 6j.
After Step 10, Γ_{11} is complete, deterministic, and compatible with R. Hence the enumeration terminates; see Example 5.1 for details of the enumerated congruence.

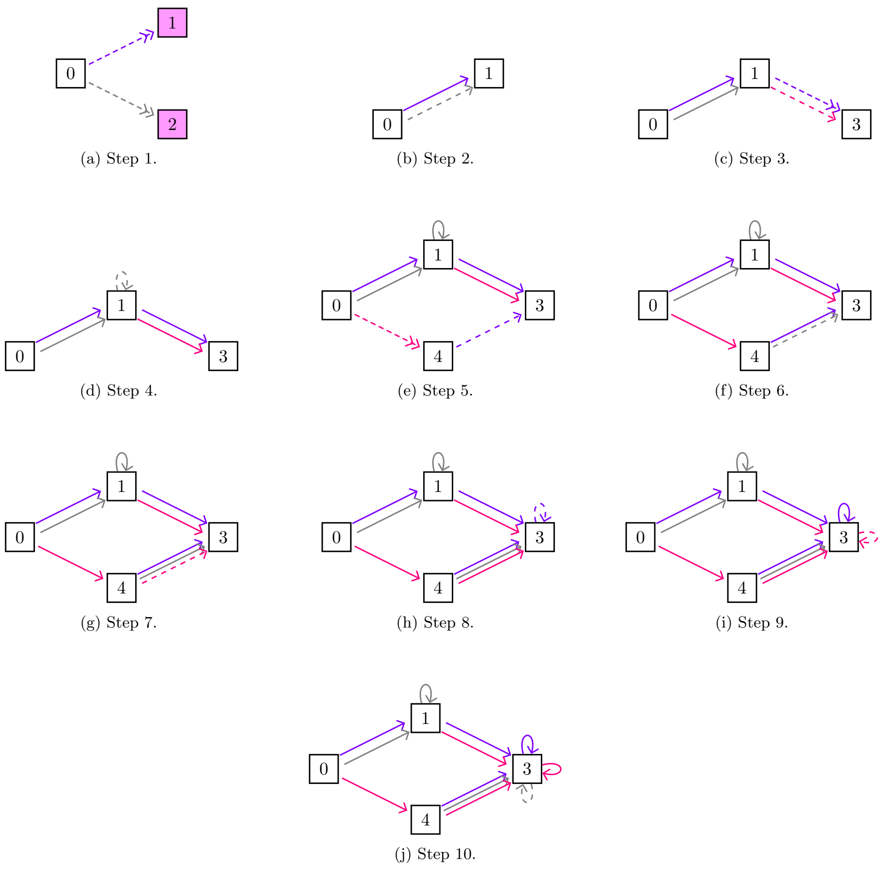


Fig. 6 The output (Γ_i, κ_i) of each step in the HLT strategy in Example 6.2. Purple arrows correspond to a, gray to b, pink to c, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes; see Table 2 for a representative word for each node. A dashed edge with a double 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

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 Γ_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 κ is trivial. The following steps are then repeatedly applied:

- **F1.** If $\alpha \in N_i$ is the minimum node in Γ_i such that there exists $a \in A$ and there is no edge with source α labelled by a, then apply **TC1** to α and a. Apply **TC2** to every node in Γ_i and every relation in R.
- **F2.** Apply **TC3** repeatedly until κ_{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

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

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 Γ_1 is the trivial graph and the input κ_1 is Δ_{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 κ_2 to be the least equivalence containing $\kappa_1 = \Delta_{N_1}$ and $(1, 2)$. The output is graph Γ_2 in Fig. 7a.

Step 2 We continue with Definition 3.3(b). We apply **TC3** and we get graph Γ_3 in Fig. 7b which is the quotient of Γ_2 by κ_2. The set κ_3 is defined to be Δ_{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 Γ_4 in Fig. 7c.

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 Γ_5 in Fig. 7d.

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 Γ_6 in Fig. 7e.

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 1 and (ac, a^2), to 0 and (ca, a^2), to 0 and (c^2, a^2) and to 3 and (b^2, b).
After Step 12, Γ_{13} in Fig. 7f 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 stategy is to squeeze as much information as possible out of every definition of an edge (α, a, β) made in **F1**. The implementation of the

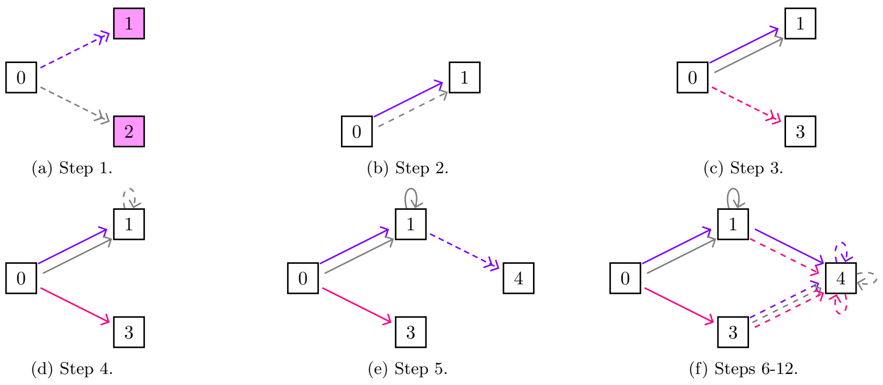


Fig. 7 The output (Γ_i, κ_i) of each step in the Felsch strategy in Example 7.1. Purple arrows correspond to a, gray to b, pink to c, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes; see Table 3 for a representative word for each node. A dashed edge with 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 3 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

0	1	2	3	4
ε	a	b	c	a^2

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 α in Γ_i if the path in Γ_i starting at α and labelled by u, or v, goes through a part of Γ_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 Γ_i and equivalence relation κ_i, at every step of a congruence enumeration. An element (α, a) of D_i corresponds to a recently defined edge incident to α and labelled by a. More precisely, we define $D_1 = \varnothing$ 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 $\langle A \mid R\rangle$ is a finite monoid presentation, then we define the *Felsch tree* $F(A, R)$ of this presentation to be a pair (Θ, ι) where:

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

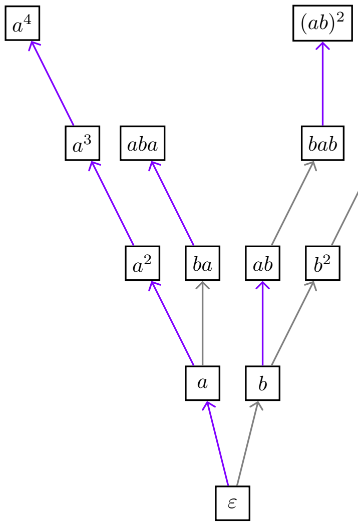


Fig. 8 The Felsch tree $F(A, R)$ from Example 7.3

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 Θ 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 Θ can found in Fig. 8.

The function $\iota : N \longrightarrow \mathfrak{P}(R)$ is:

$$\begin{aligned}
(\varepsilon)\iota &= R,\\
(a)\iota = (a^2)\iota &= \{(a^4, a), ((ab)^2, a^2)\}\\
(a^3)\iota = (a^4)\iota &= \{(a^4, a)\}\\
(aba)\iota &= \{((ab)^2, a^2)\}\\
(b)\iota = (b^2)\iota = (b^3)\iota &= \{(b^3, b)\}\\
(ab)\iota = ((ab)^2)\iota &= \{((ab)^2, a^2)\}\\
(ba)\iota = (bab)\iota &= \varnothing.
\end{aligned}$$

We modify **TC1**, **TC2**, and **TC3** so that the set D_i is defined appropriately at every step, and then, roughly speaking, we replace **F2** in the Felsch strategy by a backtrack search through Θ in $F(A, R)$ for every pair in D_i. We will refer to this as the *modified Felsch strategy*.

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

PD1. Apply **TC2** to α and every relation in $(v)\iota$.
PD2. For every edge (v, b, bv) in Θ that has not been traversed: for every node β in Γ_i such that $(\beta, b, \alpha) \in E_i$, apply **PD1** and **PD2** to β and bv.
PD3. Repeatedly apply **TC3** until κ 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 Θ, then a does not occur in any relation in R and so following the path in Γ_i labelled by any u such that $(u, v) \in R$ from any node in w in Γ_i cannot contain any edge labelled by a. In particular, no such path contains the edge $(\alpha, a, \beta) \in \Gamma_i$; the definition of which caused (α, a) to belong to D_i in the first place. Hence if $(\alpha, a) \in D_i$, then there is a node a in Θ.

We repeatedly apply **PD1**, **PD2**, and **PD3** starting from every pair in D_i. This must terminate eventually because Θ is finite and we only apply **TC2** and **TC3** in **PD1**, **PD2**, and **PD3**.

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

Lemma 7.4 *Suppose that α is a node in Γ_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 Γ_i starting at α. 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 **PD1** and **PD2** will be applied to (β, a) at some step $j \geq i$ of the enumeration. This search begins with **PD1**, where **TC2** is applied to β and every relation in $(a)\iota$. At step **PD2** we apply **PD1** and **PD2** to every $\zeta \in N_i$ and w such that there exists an edge of the form (a, b, w) in Θ and an edge $(\zeta, b, \beta) \in E_i$.

By definition, $u = b_1 \cdots b_{t-1} a b_{t+1} \cdots b_r$. If $m \in \{1, \ldots, t\}$, then, since $(u, v) \in R$, $b_m \cdots b_{t-1}a$ and $b_{m-1} \cdots b_{t-1}a$ are nodes of Θ and there exists an edge $(b_m \cdots b_{t-1}a, b_{m-1}, b_{m-1} \cdots b_{t-1}a)$ in Θ. Also by assumption $(\mu_{m-1}, b_{m-1}, \mu_m) \in E_i$ and so $(\mu_{m-1}, b_{m-1} \cdots b_{t-1}a)$ is one of the pairs in **PD2** if it is applied to $(\mu_m, b_m \cdots b_{t-1}a)$. If $m = t$, then $(\mu_m, b_m \cdots b_{t-1}a) = (\mu_t, a) = (\beta, a)$ and hence **PD2** is applied to $(\mu_m, b_m \cdots b_{t-1}a)$ for every $m \in \{1, \ldots, t\}$. In particular, **PD1** is applied to $(\mu_1, b_1 \cdots b_{t-1}a) = (\alpha, b_1 \cdots b_{t-1}a)$ at some point. This means that **TC2** is applied to α and every relation in $(b_1 \cdots b_{t-1}a)\iota$. Since $b_1 \cdots b_{t-1}a = b_1 \cdots b_t$ is a prefix of u, it follows that $(u, v) \in (b_1 \cdots b_{t-1}a)\iota$ and so **TC2** is applied to α and (u, v) at some step $j \geq i$. Because u and v were assumed to label paths originating at α, neither **TC2**(a) nor (b) applies. Additionally, we assumed that $\mu_{r+1} \neq \nu_{s+1}$ and so **TC2**(c) is applied and $(\mu_{r+1}, \nu_{s+1}) \in \kappa_{i+1}$.

As noted above, Θ is finite, and **PD1** and **PD2** are only applied finitely many times before there is an application of **PD3**. If the first application of **TC3** (in **PD3**) after step j occurs at step k, then Γ_{k+1} is the quotient of Γ_k by κ_k and $\kappa_i \subseteq \kappa_k$. Therefore $(u, v) \in (\alpha)\pi_{\Gamma_{k+1}}$ as required. □

Proposition 7.5 *If $\langle A | R\rangle$ 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 ρ is the least right* *congruence on A^* containing S and $R^\#$, then the modified Felsch strategy applied to (Γ_1, κ_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 **F1**, **TC2** 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 **TC1** are larger than all previous nodes). Hence it suffices to prove that if α is a node for all $j \geq i$, there exists some k such that $(u, v) \in (\alpha)\pi_{\Gamma_k}$. Since **F1** is repeatedly applied there exists a step $k + 1$ when α is a node in Γ_{k+1} and paths P_u and P_v leaving α labelled by u and v, respectively, exist in Γ_{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 (β, a, γ) 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 Γ_k starting at α, which contradicts the assumed minimality of k. $\square$

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 definition 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 κ_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 κ_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 α is the inverse Ackermann function; see [34].

Another issue that arises in the implementation is how to represent the word graphs Γ_i. In order to efficiently obtain the word graph Γ_{i+1} from Γ_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 $\Gamma = (N, E)$ ouput 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 (α, a, β) is an edge. As such if it ever arises that there are edges (α_1, a, β) and (α_2, a, β) in a word graph during a coset enumeration, the pair (α_1, α_2) can immediately be added to κ_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 κ_i are merged successively. A balance must be struck between repeatedly updating the data structure for the edges with given target in Γ_{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 Γ_i is comparable to the number in Γ_{i+1}, i.e. only a relatively small number of nodes are merged. On the other hand, if Γ_{i+1} is considerably smaller than Γ_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 ε is the $\preceq$-minimum word in A^* for every reduction ordering on A^*.

If the set A is totally ordered by $\leq$, then we may extend $\leq$ to $\leq_{\text{lex}}$ over A^*, by defining $\varepsilon \leq_{\text{lex}} w$ for all $w \in A^*$ and $u \leq_{\text{lex}} v$ whenever $u = au_1$ and $v = bv_1$ for some $a, b \in A$ with $a < b$, or $a = b$ and $u_1 \leq_{\text{lex}} v_1$. 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* $\leq_{\text{slex}}$ on A^* is defined as follows: if $u, v \in A^*$, then $u \leq_{\text{slex}} v$ if $|u| < |v|$ or $|u| = |v|$ and $u \leq_{\text{lex}} v$. 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 $w_\alpha \prec w_\beta$ 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 Γ into a standardized word graph, is referred to as *standardization*; for example, see STANDARDIZE in [31, p195].

A word graph Γ_i can be replaced by any standardized word graph at any step of a congruence enumeration, provided that the values in κ_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 Γ_{i+1} in both **HLT1** and **F1** depends on the numerical values of their source nodes. Hence replacing Γ_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

$$\langle a, b, 0 \mid ab = 0, a^4 = a, b^3 = b, (ab)^2 = 0, a0 = 0a = 0 = b0 = 0b = 0^2\rangle,$$

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 $\langle A \mid R\rangle$ 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 ω 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 β 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 (Γ_1, κ_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 $\langle A \mid R\rangle$ 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 $\langle A \mid R\rangle$ 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 Γ for some $\alpha \in N$, then the aim of this variant is to output the subgraph Λ of Γ induced by the set X of nodes in N from which α is reachable. Note that the set of these nodes corresponds to the set of elements in M which are $\geq_{\mathcal{R}} w/R^\#$ (recall that two monoid elements $s, t \in M$ satisfy $s \geq_{\mathcal{R}} t$ if $tM \subseteq sM$). If $\mathcal{A}$ is the automata with alphabet A, state set X, initial state 0, accept state α, and edges consisting of those in Λ, then the language $L(\mathcal{A})$ accepted by $\mathcal{A}$ 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 Γ_n consists of the single path from 0 to the node n. (The graph Γ_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 α labelled by v_1 and then **TC2** is applied to α and $(u, v) \in R$. (This is referred to as an *elementary expansion* in [32].)

S3: Apply **TC3**. (Quotienting Γ_i by the least equivalence containing a single $(\alpha, \beta) \in N_i \times N_i$ is referred to as a *determination* in [32]. In **TC3** we quotient Γ_i by the entire equivalence κ_i; this is the only point where the procedure described here differs from the description 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 Γ_i is isomorphic to the induced subgraph Λ defined at the start of this section. Note that **S2** and **S3** are similar to **HLT1**, **HLT2**, and **HLT3** described in Sect. 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 $\langle A \mid R\rangle$ 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 Λ is finite. Since the set of nodes in Λ corresponds to the set of elements in M which are $\geq_{\mathcal{R}} u/R^{\#}$ (assuming that u is the input word for Stephen's procedure) the word graph Λ is finite when there are only finitely many elements of M that are $\geq_{\mathcal{R}} u/R^{\#}$.

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

$$\mathcal{P} = \langle a, b \mid a^3 = a,\ b^3 = b,\ (ab)^2 = a^2\rangle$$

with $a < b$. The input word graph Γ_1 is the trivial graph and the input $\kappa_1 = \Delta_{N_1}$. Since $S = \varnothing$, 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 **F1** (a single application of **TC1** and multiple applications of **TC2**) or a single application of **TC3**. If a step produces no change to Γ_i or κ_i, this step is skipped and does not have a number; see Table 4 and Figs. 9, 10, 11, 12, and 13.

Step 1–2 The only node in Γ_1 is 0. Since there is no edge incident to 0 labelled by a, in Step 1 we apply **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 Γ_3 in Fig. 9a.

Step 3 An application of **TC1** which leads to the definition of the node 3 and the edge $(1, a, 3)$. At this point **F1** leads to an application of **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 Γ_4; see Fig. 9b.

Step 4–5 We apply **TC1** twice (since there are no applications of **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. 9c.

Step 6 The node 6 and the edge $(2, b, 6)$ are defined. Applying **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 Γ_7; see Fig. 9d.

Step 7 We apply **TC1** and define the node 7 and the edges $(3, b, 7)$; see Fig. 9e.

Step 8 The node 8 and edge $(4, a, 8)$ are defined in **TC1**. Applying **TC2** leads to the definition of edges $(7, b, 3)$, $(8, b, 3)$, and $(8, a, 4)$; see Fig. 9f.

Step 9 We apply **TC1** and define the node 9 and the edge $(4, b, 9)$. We apply **TC2** to 4 and the relation (b^3, b) and define the edge $(9, b, 4)$; see Fig. 9g.

Step 10 We apply **TC1** and define the node 10 and the edge $(5, a, 10)$. We apply **TC2** to the node 5 and the relation (a^3, a) and define the edge $(10, a, 5)$; see Fig. 9h.

Step 11–12 We apply **TC1** twice, since there are no applications of **TC2** yielding any new information. The nodes 11 and 12 and the edges $(5, b, 11)$ and $(6, a, 12)$ are defined; see Fig. 9i.

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

Step 14 In this step κ_{14} is the least equivalence relation containing $\{(1, 9), (4, 13)\}$. Taking the quotient gives graph Γ_{15}. In Γ_{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 κ_{15}; see Fig. 10a.

Step 15 Taking the quotient of Γ_{15} in **F2** by κ_{15} yields the graph in Fig. 10b.

Step 16 We apply **TC1** and define the node 14 and edge $(10, b, 14)$; see Fig. 10c.

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

Step 18 The node 16 and the edge $(11, b, 16)$ are defined. After an application of **TC2** to 5 and (b^3, b) leads to the definition of $(16, b, 11)$; see Fig. 10e.

Step 19 The node 17 and the edge $(12, a, 17)$ are defined. An application of **TC2** to 6 and (b^3, b) leads to the definition of $(17, a, 12)$; see Fig. 10f.

Step 20 We apply **TC1** and the node 18 and the edge $(12, b, 18)$ are defined; see Fig. 11a.

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

Table 4 A word labelling a path from 0 to each node in Example A.1; see Figs. 9, 10, 11, 12, and 13

0	1	2	3	4	5	6	7	8	9	10	11
ε	a	b	a^2	ab	ba	b^2	a^2b	aba	ab^2	ba^2	bab
12	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**
b^2a	a^2ba	ba^2b	$(ba)^2$	bab^2	b^2a^2	b^2ab	ba^2ba	b^2a^2b	b^2aba	b^2ab^2	b^2a^2ba

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. 11c.

Step 23 Taking the quotient gives graph Γ_{24}; see Fig. 11d.

Step 24 We apply **TC1** so that the node 20 and the edge $(17, b, 20)$ are defined; see Fig. 12a.

Step 25 We apply **TC1** and the node 21 and the edge $(18, a, 21)$ are defined. We apply **TC2** to 6 and $((ab)^2, a^2)$ and define the edge $(21, b, 17)$. We also apply **TC2** to 18 and $((ab)^2, a^2)$ and define $(21, a, 18)$. Finally, we apply **TC2** to 6 and (b^3, b) and we define $(20, b, 17)$; see Fig. 12b.

Step 26 We apply **TC1** and define 22 and $(18, b, 22)$. Next, we apply **TC2** to 12 and (b^3, b). This leads to the definition of $(22, b, 18)$; see Fig. 12c.

Step 27 We apply **TC1** and define 23 and $(20, a, 23)$. We apply **TC2** to 12 and $((ab)^2, a^2)$ and define $(23, b, 12)$. We also apply **TC2** to 20 and $((ab)^2, a^2)$ and define $(23, a, 20)$. Finally, we apply **TC2** to 23 and (b^3, b) and to 23 and $((ab)^2, a^2)$ yielding the pairs $(12, 22)$ and $(18, 23)$ in κ_{28}; see Fig. 12d.

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 b^2ab and hence the pair $(20, 21)$ is added to κ_{29}; see Fig. 13a.

Step 29 Taking the quotient of Γ_{29} by κ_{29} gives the graph Γ_{30}; see Fig. 13b. 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:

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

with $a < b < c$. Each step of this enumeration corresponds to either: at least one application of **TC1** (in **HLT1**) followed by multiple applications of **TC2** (in the same **HLT1** step as the application of **TC1** or subsequent applications of **HLT1** where **TC1** is not invoked); or a single application of **TC3** (in **HLT2**); see Figs. 14 and 15 for the word graphs Γ_i and equivalence relation κ_i after every step i; see also Table 5.

Step 1 At this step we apply **HLT1** to the node 0 and relation (a^2, ac). **TC1** yields the new nodes 1 and 2, and the edges $(0, a, 1)$ and $(1, a, 2)$; **TC2**(b) then yields the edge $(1, c, 2)$; see Fig. 14a.

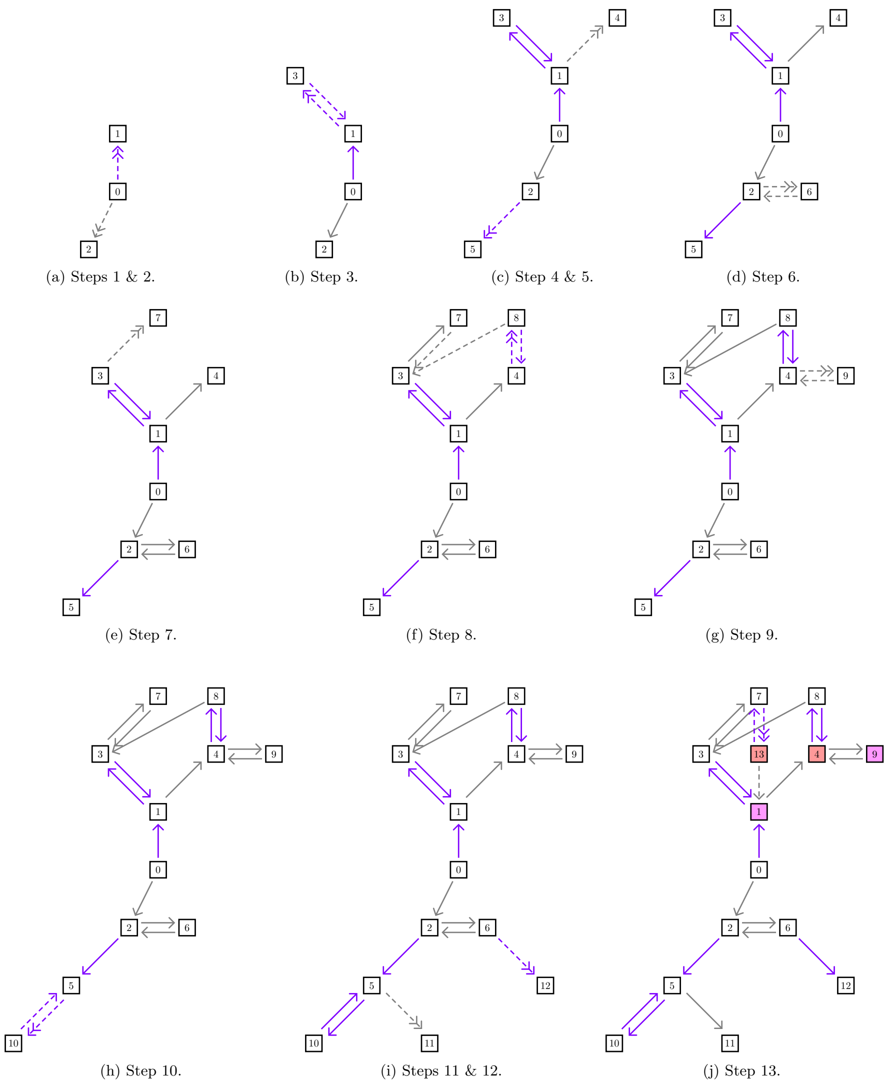


Fig. 9 The output (Γ_i, κ_i) for $i = \{3, 4, 6, 7, 8, 9, 10, 11, 13, 14\}$ in the Felsch Strategy in Example 7.1. Purple arrows correspond to a, gray to b, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the 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

Step 2 We apply **HLT1** to the node 0 and relation (b^2, b). **TC1** produces the new node 3 and edge $(0, b, 3)$; **TC2**(a) then yields the edge $(3, b, 3)$; see Fig. 14b.

Step 3 **HLT1** is applied to 0 and (a^2, ca); **TC1** generates the node 4 and edge $(0, c, 4)$; **TC2**(b) gives us the edge $(4, a, 2)$; see Fig. 14c.

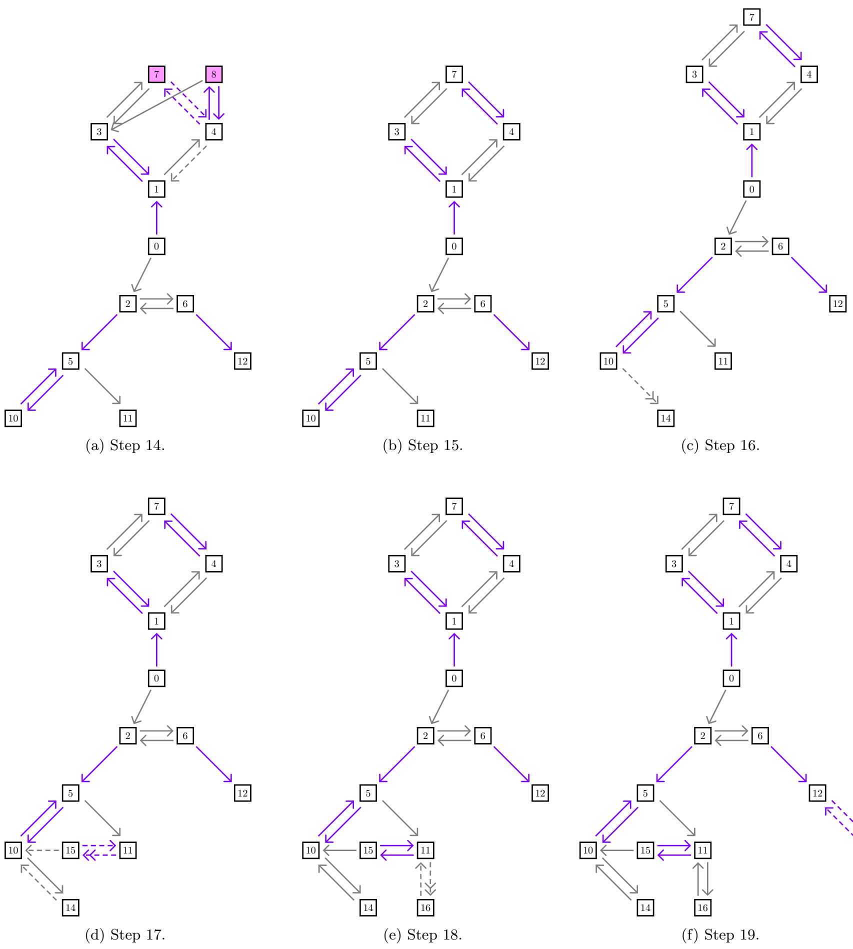
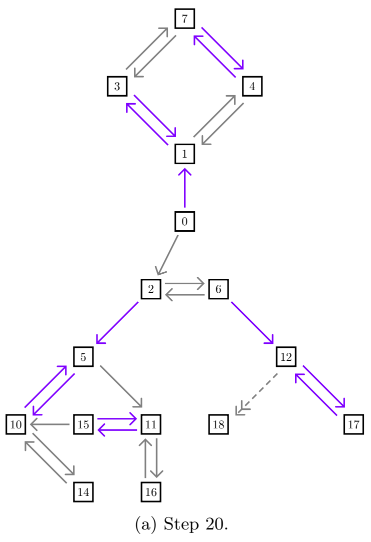


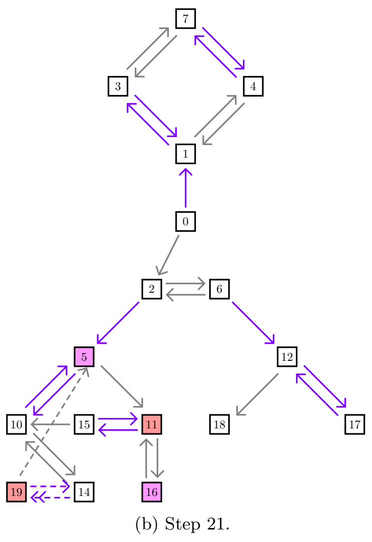
Fig. 10 The output (Γ_i, κ_i) for $i \in \{15, \dots, 20\}$ in the Felsch strategy in Example 7.1. Purple arrows correspond to a, gray to b, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined 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

Steps 4–6 Step 4 is **HLT1** applied to 0 and (bc, cb); **TC1** gives the new node 5 and edge $(3, c, 5)$; **TC2**(b) yields the edge $(4, b, 5)$. Step 5 is **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 **TC1** is not invoked, but **TC2**(b) yields the edge $(4, c, 2)$. Step 6: **HLT1** applied to 0 and (a^3, a^2), **TC1** is not invoked again, and **TC2**(a) yields the edge $(2, a, 2)$; see Fig. 14d.

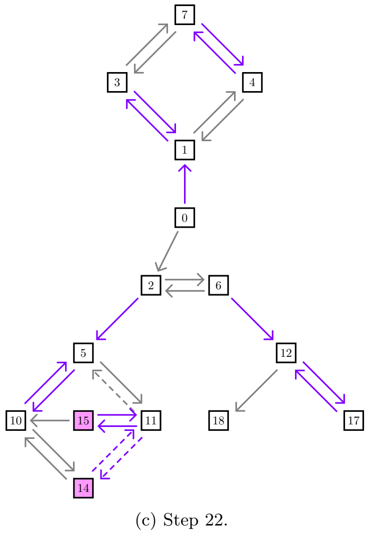
Steps 7–9 Step 7: **HLT1** applied to 0 and (a^2, aba); **TC1** adds the node 6 and edge $(1, b, 6)$; **TC2**(b) yields $(6, a, 2)$. Step 8: **HLT1** applied to 1 and (a^2, ac); **TC1** does not apply; **TC2**(b) yields $(2, c, 2)$. Step 9: **HLT1** applied to 1 and (b^2, b); **TC1** does not apply; **TC2**(a) yields $(6, b, 6)$. See Fig. 14e.



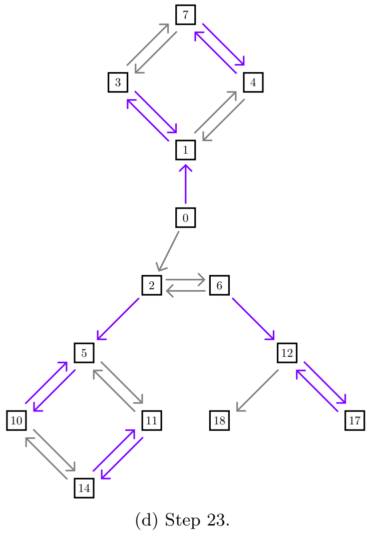
(a) Step 20.



(b) Step 21.



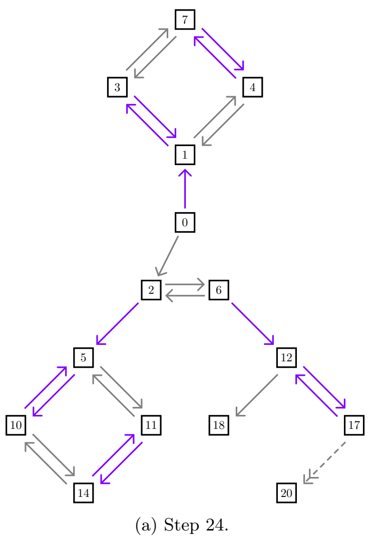
(c) Step 22.

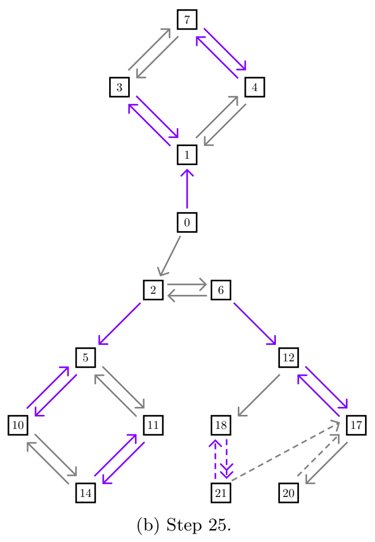


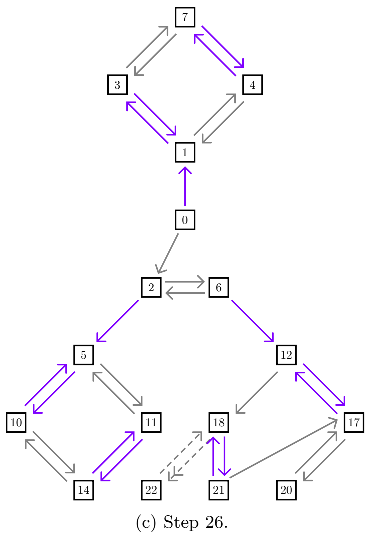
(d) Step 23.

Fig. 11 The output (Γ_i, κ_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 κ_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

Steps 10–13 Step 10: **HLT1** applied to 1 and (bc, cb); **TC1** adds node 7 and edge $(6, c, 7)$. Step 11: **HLT1** applied to 1 and (a^2, aba); **TC1** does not apply; **TC2**(b) yields edge $(7, a, 2)$. Step 12: **HLT1** applied to 2 and (b^2, b); **TC1** does not apply; **TC2**(a) yields edge $(7, b, 7)$. Step 13: **HLT1** applied to 2 and (bc, cb); **TC1** does not apply; **TC2**(a) yields edge $(7, c, 7)$. See Fig. 14f.







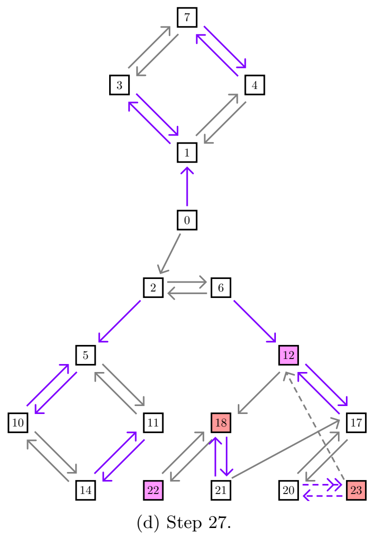
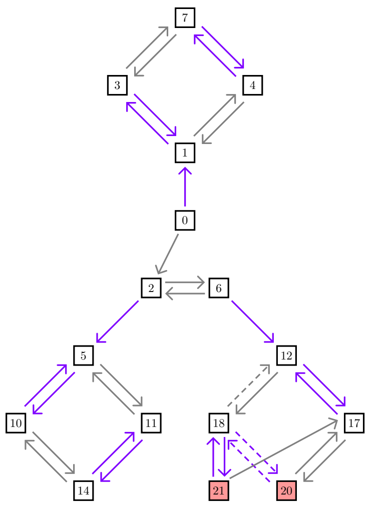


Fig. 12 The output (Γ_i, κ_i) for $i \in \{25, \ldots, 28\}$ in the Felsch strategy for Example 7.1. Purple arrows correspond to a, gray to b, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being 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

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


(a) Step 28.

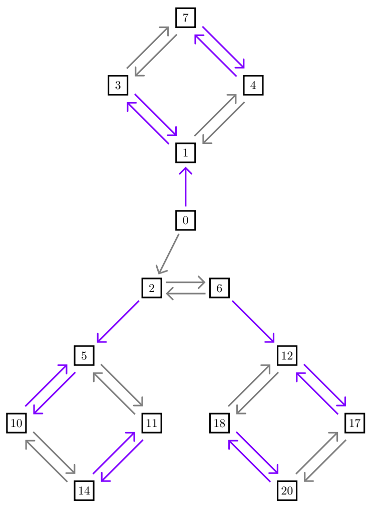

(b) Step 29.

Fig. 13 The output (Γ_i, κ_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 κ_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

Steps 19–20 Step 19: **HLT1** for 3 and (a^2, aba): **TC1** adds node 10 and edge $(8, b, 10)$; **TC2**(b) yields $(10, a, 9)$. Step 20: **HLT1** for 4 and (bc, cb): **TC1** does not apply; **TC2**(c) indicates κ_{20} is the least equivalence containing $(7, 9)$. See Fig. 14h.

Step 21 This step is an application of **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 Γ_{21} meaning that κ_{21} is the least equivalence containing $(2, 7)$. See Fig. 15a.

Step 22 This step is also an application of **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. 15b.

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. 15c.

After Step 24 the graph Γ_{25} is complete and compatible with the relations in the presentation, and κ_{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:

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

Table 5 A word labelling a path from 0 to each node in the right Cayley graph on the monoid M from Example A.2; see Figs. 14 and 15

0	1	2	3	4	5	6	7	8	9	10
ε	a	a^2	b	c	bc	ab	abc	ba	ba^2	bab

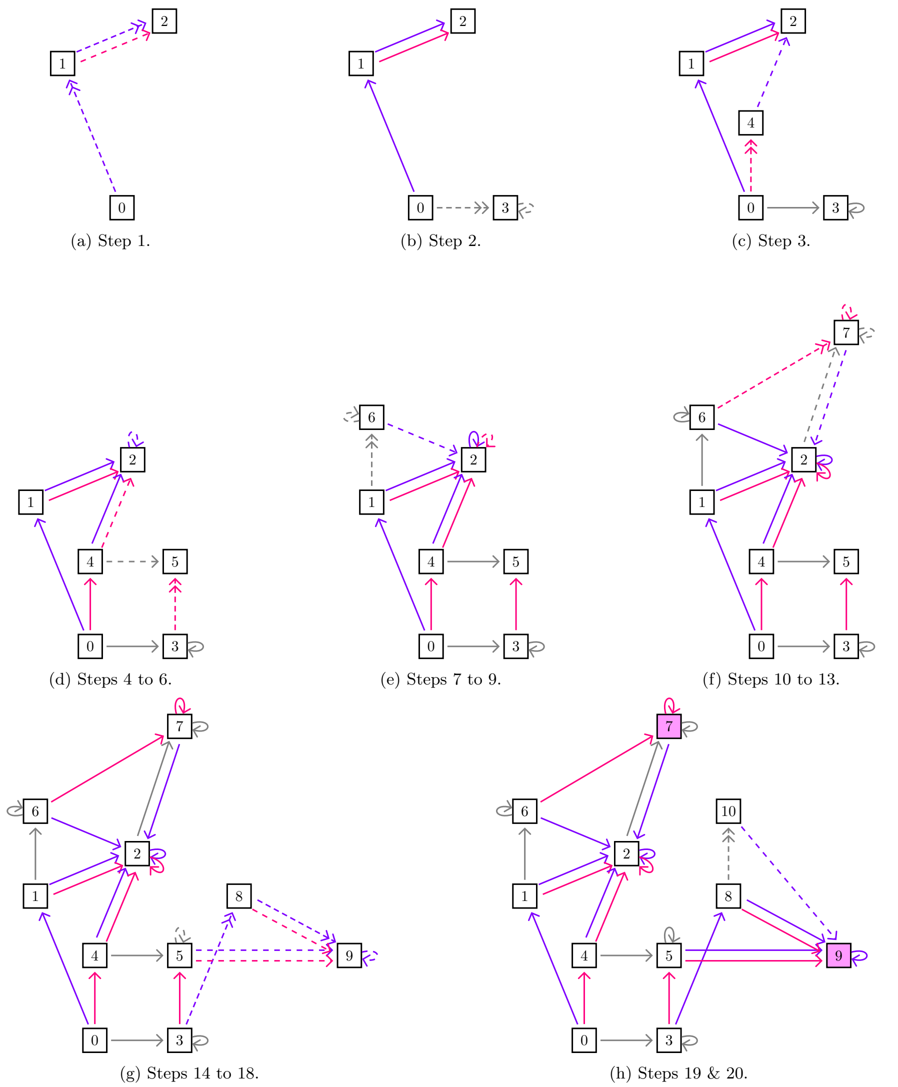


Fig. 14 The output (Γ_i, κ_i) for $i = 1, \ldots, 20$ in Example A.2. Purple arrows correspond to a, gray to b, pink to c, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined in **TC1**, 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

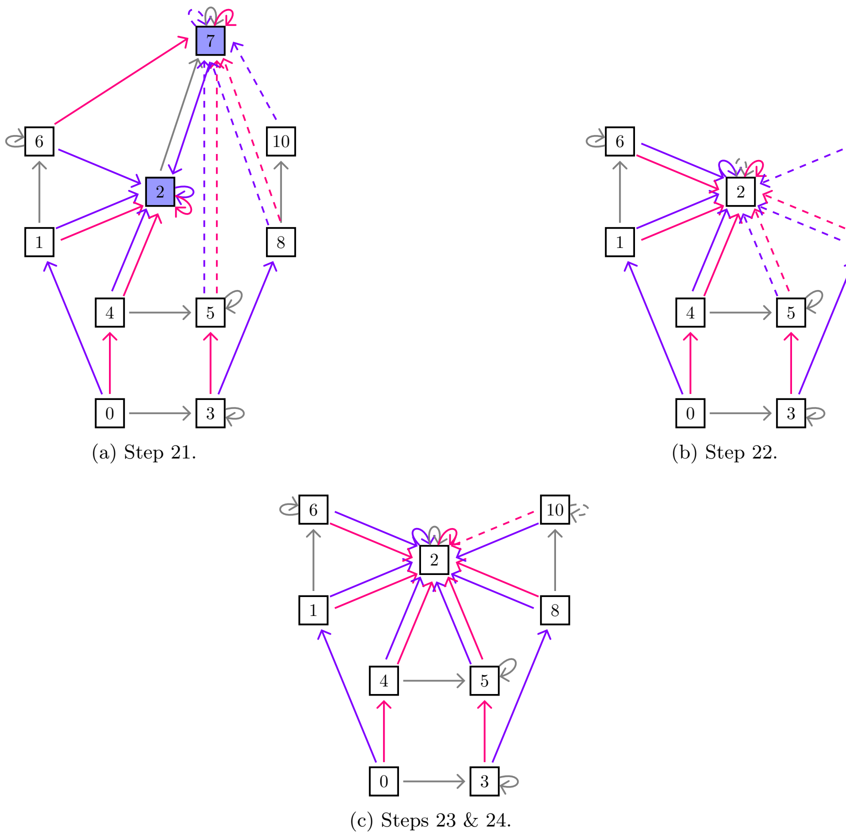


Fig. 15 The output (Γ_i, κ_i) for $i = 21, \ldots, 24$ of each step in Example A.2. Purple arrows correspond to 1, gray to 3, pink to 4, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes. A dashed edge with a double arrowhead indicates the edge being defined 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

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. 16a.

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. 16b.

Step 3 We apply **F1** to 0. The node 3 and the edge $(0, c, 3)$ are defined; see Fig. 16c.

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, ca) yields the edge $(3, a, 4)$, 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. 16d.

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. 16e.

Step 6 We apply **F1** to 2 which leads to the definition of the node 6 and the edge $(2, a, 6)$; see Fig. 16f.

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. 17a.

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 κ_9 is the least equivalence containing $(8, 4)$ and Δ_{N_9}; see Fig. 17b.

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. 17c.

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 κ_{11} is the least equivalence containing $(4, 9)$ and $\Delta_{N_{11}}$; see Fig. 17d.

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. 17e.

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. 17f.

After Step 12 the graph Γ_{13} is complete and compatible with the relations in the presentation, and κ_{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.

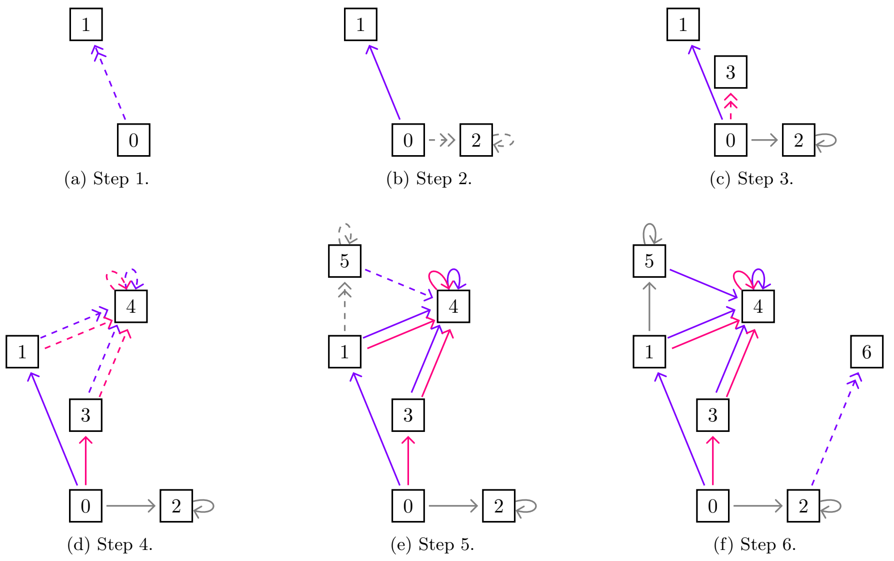


Fig. 16 The output (Γ_i, κ_i) of each step in Example A.3. Purple arrows correspond to a, grey to b, pink to c, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes

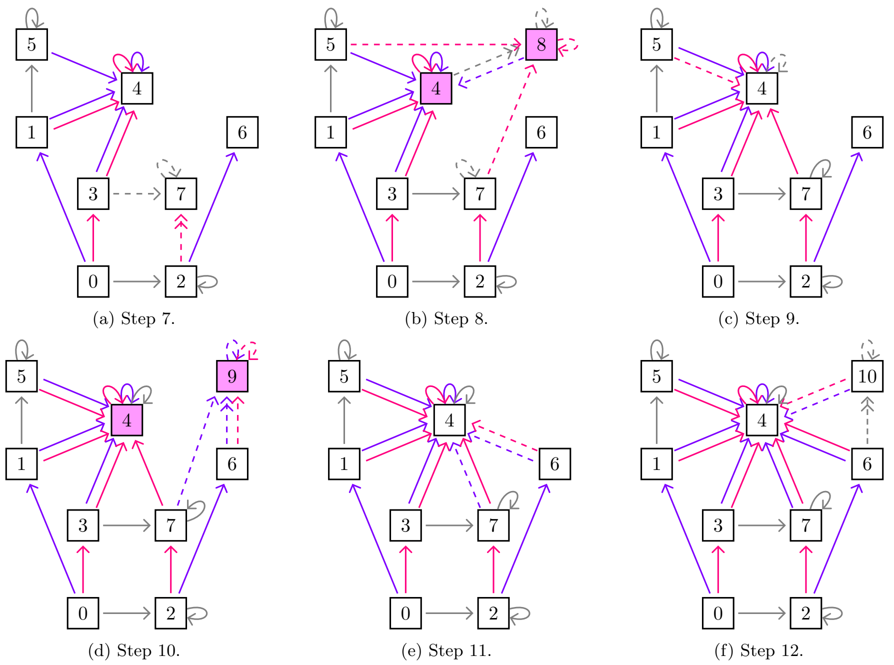


Fig. 17 The output (Γ_i, κ_i) of each step in Example A.3. Purple arrows correspond to a, grey to b, pink to c, shaded nodes of the same colour belong to κ_i, and unshaded nodes belong to singleton classes

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 16 GB 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 6 and 7.

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 8, 9, 10, 11, and 12.

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 13, 14, and 15.

The presentations referred to in Table 15 are:

$$\begin{aligned} S = \langle a, b, c \mid\ & a^{14} = a, b^{14} = b, c^{14} = c, a^4ba = b^3, b^4ab = a^3, \\ & a^4ca = c^3, c^4ac = a^3, b^4cb = c^3, c^4bc = b^3\rangle \end{aligned} \tag{B.1}$$

$$S = \langle a, b \mid\ a^{32} = a, b^3 = b, ababa = b, a^{16}ba^4ba^{16}ba^4\rangle \tag{B.2}$$

$$S = \langle a, b \mid\ a^{16} = a, b^{16} = b, ab^2 = ba^2\rangle \tag{B.3}$$

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

$$S = \langle a, b \mid\ a^3 = a, b^6 = b, (abab^4)^7ab^2ab^5a^2 = b^2\rangle \tag{B.5}$$

$$S = \langle a, b \mid\ a^3 = a, b^9 = b, abab^7abab^7ab^2ab^8 = b^2\rangle \tag{B.6}$$

$$\begin{aligned} S = \langle a, b, c, d, e \mid\ & a^3 = a, b^3 = b, c^3 = c, d^3 = d, \\ & e^3 = 3, (ab)^3 = a^2, (bc)^3 = b^2, (cd)^3 = c^2, (de)^3 = d^2, \\ & ac = ca, ad = da, ae = ea, bd = db, be = eb, ce = ec\rangle \end{aligned} \tag{B.7}$$

$$S = \langle a, b \mid\ a^3 = a, b^{23} = b, ab^{11}ab^2 = b^2a\rangle. \tag{B.8}$$

Table 6 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]

n	$\|OP_n\|$	$\|A\|$	$\|R\|$	$\sum \|uv\|$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
3	24	2	5	43	8.4×10^{-6}	8×10^{-7}	1.84×10^{-5}	1.8×10^{-6}	5.3×10^{-4}	7×10^{-5}
4	128	2	6	86	4.0×10^{-5}	2×10^{-6}	2.77×10^{-4}	5×10^{-6}	1.92×10^{-3}	7×10^{-5}
5	610	2	7	145	4.51×10^{-4}	4×10^{-6}	4.565×10^{-3}	1.5×10^{-5}	1.492×10^{-2}	1.3×10^{-4}
6	2742	2	8	220	4.64×10^{-3}	7×10^{-5}	4.153×10^{-2}	1.2×10^{-4}	1.174×10^{-1}	3×10^{-4}
7	11,970	2	9	311	4.97×10^{-2}	4×10^{-4}	4.0×10^{-1}	5×10^{-2}	8.081×10^{-1}	1.1×10^{-3}
8	51,424	2	10	418	7.11×10^{-1}	5×10^{-3}	3.07×10^{0}	4×10^{-2}	4.488×10^{0}	2×10^{-3}
9	218,718	2	11	541	5.40×10^{0}	3×10^{-2}	2.98×10^{1}	2×10^{-1}	–	–
10	923,690	2	12	680	2.503×10^{1}	–	–	–	–	–
11	3,879,766	2	13	835	2.680×10^{2}	–	–	–	–	–
12	16,224,804	2	14	1006	2.351×10^{3}	–	–	–	–	–
13	67,603,744	2	15	1193	3.160×10^{4}	–	–	–	–	–

Values with no standard deviation indicated were only run once. All times are in seconds

Table 7 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]

n	$\lvert OR_n\rvert$	$\lvert A\rvert$	$\lvert R\rvert$	$\sum \lvert uv\rvert$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
3	27	3	9	65	1.388×10^{-5}	9×10^{-8}	3.75×10^{-5}	3×10^{-7}	6.78×10^{-4}	1.2×10^{-5}
4	180	3	10	122	8.45×10^{-5}	6×10^{-7}	5.784×10^{-4}	1.7×10^{-6}	4.18×10^{-3}	1.5×10^{-4}
5	1015	3	11	199	1.120×10^{-3}	1.0×10^{-5}	1.034×10^{-2}	5×10^{-5}	3.140×10^{-2}	8×10^{-5}
6	5028	3	12	296	1.257×10^{-2}	6×10^{-5}	1.459×10^{-1}	3×10^{-4}	2.466×10^{-1}	2×10^{-4}
7	23,051	3	13	413	2.404×10^{-1}	1.2×10^{-3}	1.369×10^{0}	5×10^{-3}	1.8565×10^{0}	1.1×10^{-3}
8	101,272	3	14	550	4.94×10^{0}	3×10^{-2}	1.437×10^{1}	4×10^{-2}	1.302×10^{1}	4×10^{-2}
9	434,835	3	15	707	7.035×10^{0}	–	–	–	–	–
10	1,843,320	3	16	884	8.509×10^{1}	–	–	–	–	–
11	7,753,471	3	17	1081	8.460×10^{2}	–	–	–	–	–
12	32,440,884	3	18	1298	6.178×10^{3}	–	–	–	–	–

Values with no standard deviation indicated were only run once. All times are seconds

Table 8 Comparison of libsemigroups [24] and the implementation in the GAP [12] library (`CosetTableOfFpSemigroup`) for the presentations of the partition monoids P_n from [8, Theorem 41]

n	$\lvert P_n \rvert$	$\lvert A \rvert$	$\lvert R \rvert$	$\sum \lvert uv \rvert$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
4	4140	4	19	144	7.43×10^{-3}	1.2×10^{-4}	1.918×10^{-2}	5×10^{-5}	1.2919×10^{-1}	1.1×10^{-4}
5	115,975	4	19	163	5.69×10^{-1}	2×10^{-3}	1.082×10^{0}	7×10^{-3}	5.377×10^{0}	6×10^{-3}
6	4,213,597	4	20	198	3.24×10^{1}	3×10^{-1}	1.460×10^{2}	5×10^{-1}	–	–
7	190,899,322	4	20	219	2.240×10^{3}	–	–	–	–	–

Values with no standard deviation indicated were only run once. All times are in seconds

Table 9 Comparison of libsemigroups [24] and the implementation in the GAP [12] library (`CosetTableOfFpSemigroup`) on the presentations for the singular Brauer monoids $B_n \setminus S_n$ from [21]

n	$\lvert B_n \setminus S_n \rvert$	$\lvert A \rvert$	$\lvert R \rvert$	$\sum \lvert uv \rvert$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
3	9	6	21	78	1.72×10^{-5}	2×10^{-7}	2.4×10^{-5}	4×10^{-6}	4.6×10^{-4}	4×10^{-5}
4	81	12	126	600	1.955×10^{-4}	8×10^{-7}	6.27×10^{-4}	1.8×10^{-5}	5.50×10^{-3}	1.2×10^{-4}
5	825	20	450	2300	5.33×10^{-3}	5×10^{-5}	1.968×10^{-2}	1.8×10^{-4}	1.60×10^{-1}	2×10^{-3}
6	9675	30	1185	6240	1.359×10^{-1}	2×10^{-4}	6.59×10^{-1}	6×10^{-3}	4.81×10^{0}	4×10^{-2}
7	130,095	42	2583	13,818	3.89×10^{0}	2×10^{-2}	2.190×10^{1}	1.2×10^{-1}	–	–
8	1,986,705	56	4956	26,768	3.357×10^{2}	–	–	–	–	–

Values with no standard deviation indicated were only run once. All times are in seconds

Table 10 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]

n	$\lvert I_n^* \rvert$	$\lvert A \rvert$	$\lvert R \rvert$	$\sum \lvert uv \rvert$	HLT		Felsch		Rc		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D	Mean	S.D
3	25	3	11	54	1.5×10^{-5}	5×10^{-6}	1.92×10^{-5}	5×10^{-7}	1.850×10^{-5}	8×10^{-8}	4.94×10^{-4}	1.0×10^{-5}
4	339	4	19	116	4.17×10^{-4}	4×10^{-6}	6.05×10^{-4}	4×10^{-6}	1.104×10^{-3}	9×10^{-6}	8.74×10^{-3}	2×10^{-5}
5	6721	5	28	177	3.15×10^{-2}	3×10^{-4}	2.780×10^{-2}	7×10^{-5}	2.926×10^{-2}	6×10^{-5}	3.213×10^{-1}	5×10^{-4}
6	179,643	6	39	276	3.639×10^{0}	9×10^{-3}	1.399×10^{0}	1.2×10^{-2}	4.044×10^{0}	1.3×10^{-2}	3.858×10^{1}	4×10^{-2}
7	6,166,105	7	52	422	–	–	1.115×10^{2}	–	–	–	–	–

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

Table 11 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]

n	$\|FI_n^*\|$	$\|A\|$	$\|R\|$	$\sum \|uv\|$	HLT		Felsch		Rc		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D	Mean	S.D
3	16	3	8	47	1.1×10^{-5}	4×10^{-6}	2.6×10^{-5}	4×10^{-6}	1.57×10^{-5}	1.4×10^{-6}	4.48×10^{-4}	1.7×10^{-5}
4	131	4	13	67	8.8×10^{-5}	4×10^{-6}	1.54×10^{-4}	1.1×10^{-5}	1.02×10^{-4}	4×10^{-6}	2.302×10^{-3}	1.3×10^{-5}
5	1496	5	20	95	2.64×10^{-3}	5×10^{-5}	2.243×10^{-3}	2.0×10^{-5}	3.35×10^{-3}	1.0×10^{-4}	3.612×10^{-2}	5×10^{-5}
6	22,482	6	29	131	7.98×10^{-2}	4×10^{-4}	5.39×10^{-2}	4×10^{-4}	5.67×10^{-2}	2×10^{-4}	8.186×10^{-1}	1.3×10^{-3}
7	426,833	7	40	175	4.78×10^{0}	4×10^{-2}	1.608×10^{0}	1.3×10^{-2}	4.162×10^{0}	1.1×10^{-2}	1.466×10^{2}	1.0×10^{0}
8	9,934,563	8	53	227	8.030×10^{1}	–	5.572×10^{1}	–	8.193×10^{1}	–	–	–

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

Table 12 Comparison of libsemigroups [24] and the implementation in the GAP [12] library (`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 *Jones monoid* in the literature

n	$\|J_n\|$	$\|A\|$	$\|R\|$	$\sum \|uv\|$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
3	5	2	4	14	4.87×10^{-6}	6×10^{-8}	5.9×10^{-6}	4×10^{-7}	2.69×10^{-4}	1.2×10^{-5}
4	14	3	8	29	8.34×10^{-6}	8×10^{-8}	1.132×10^{-5}	1.0×10^{-7}	3.26×10^{-4}	6×10^{-6}
5	42	4	13	48	1.7×10^{-5}	2×10^{-6}	2.686×10^{-5}	1.5×10^{-7}	5.39×10^{-4}	1.1×10^{-5}
6	132	5	19	71	4.18×10^{-5}	6×10^{-7}	9.51×10^{-5}	1.0×10^{-6}	1.464×10^{-3}	1.1×10^{-5}
7	429	6	26	98	1.463×10^{-4}	1.4×10^{-6}	4.50×10^{-4}	4×10^{-6}	5.35×10^{-3}	6×10^{-5}
8	1430	7	34	129	5.93×10^{-4}	4×10^{-6}	2.248×10^{-3}	1.0×10^{-5}	2.274×10^{-2}	5×10^{-5}
9	4862	8	43	164	2.456×10^{-3}	9×10^{-6}	1.034×10^{-2}	5×10^{-5}	9.753×10^{-2}	1.4×10^{-4}
10	16,796	9	53	203	1.052×10^{-2}	4×10^{-5}	4.57×10^{-2}	4×10^{-4}	4.15×10^{-1}	3×10^{-3}
11	58,786	10	64	246	4.238×10^{-2}	1.4×10^{-4}	1.966×10^{-1}	1.4×10^{-3}	1.789×10^{0}	9×10^{-3}
12	208,012	11	76	293	1.805×10^{-1}	1.6×10^{-3}	8.52×10^{-1}	8×10^{-3}	8.73×10^{0}	6×10^{-2}
13	742,900	12	89	344	7.791×10^{-1}	7×10^{-4}	3.63×10^{0}	3×10^{-2}	4.40×10^{1}	4×10^{-1}
14	2,674,440	13	103	399	3.235×10^{0}	9×10^{-3}	1.517×10^{1}	1.1×10^{-1}	–	–
15	9,694,845	14	118	458	1.424×10^{1}	–	–	–	–	–
16	35,357,670	15	134	521	6.419×10^{1}	–	–	–	–	–

Values with no standard deviation indicated were only run once. All times are in seconds

Table 13 Comparison of libsemigroups [24] and the implementation in the GAP [12] library (`CosetTableOfFpSemigroup`) for the presentations of the stellar monoids from [13]

n	$\lvert \mathrm{Stellar}(n)\rvert$	$\lvert A\rvert$	$\lvert R\rvert$	$\sum \lvert uv\rvert$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
3	16	3	9	45	9.9×10^{-6}	1.2×10^{-6}	1.43×10^{-5}	3×10^{-7}	4.10×10^{-4}	1.2×10^{-5}
4	65	4	14	71	2.239×10^{-5}	1.8×10^{-7}	5.35×10^{-5}	4×10^{-7}	9.69×10^{-4}	1.4×10^{-5}
5	326	5	20	103	1.017×10^{-4}	8×10^{-7}	4.30×10^{-4}	5×10^{-6}	4.80×10^{-3}	2×10^{-5}
6	1,957	6	27	141	8.29×10^{-4}	4×10^{-6}	4.10×10^{-3}	3×10^{-5}	3.660×10^{-2}	5×10^{-5}
7	13,700	7	35	185	9.27×10^{-3}	5×10^{-5}	4.192×10^{-2}	1.2×10^{-4}	3.310×10^{-1}	3×10^{-4}
8	109,601	8	44	235	1.554×10^{-1}	6×10^{-4}	4.603×10^{-1}	1.6×10^{-3}	3.473×10^{0}	3×10^{-3}
9	986,410	9	54	291	9.67×10^{0}	1.4×10^{-1}	5.62×10^{0}	5×10^{-2}	–	–
10	9,864,101	10	65	353	–	–	7.337×10^{1}	–	–	–
11	108,505,112	11	77	421	–	–	1.018×10^{3}	–	–	–

Values with no standard deviation indicated were only run once. All times are in seconds

Table 14 Comparison of libsemigroups [24] and the implementation in the GAP [12] library (`CosetTableOfFpSemigroup`) for the presentations of the stylic monoids from [1]

n	$\lvert\mathrm{Stylic}(n)\rvert$	$\lvert A\rvert$	$\lvert R\rvert$	$\sum\lvert uv\rvert$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	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	4140	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	1032	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	1467	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	2010	2.5358×10^{0}	1.4×10^{-3}	1.625×10^{1}	9×10^{-2}	–	–
11	4,213,597	11	451	2673	2.155×10^{1}	–	–	–	–	–
12	27,644,437	12	584	3468	2.503×10^{2}	–	–	–	–	–

Values with no standard deviation indicated were only run once. All times are in seconds

Table 15 Comparison of libsemigroups [24] and the implementation in the GAP [12] library (`CosetTableOfFpSemigroup`) on examples from Walker [36]. All times are in seconds

n	$\lvert S\rvert$	$\lvert A\rvert$	$\lvert R\rvert$	$\sum \lvert uv\rvert$	HLT		Felsch		GAP	
					Mean	S.D	Mean	S.D	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 16 Comparison of libsemigroups [24], ACE [14], and the implementation in the GAP [12] library (`CosetTableFromGensAndRels`). All times are in seconds

n	$\|G:H\|$	$\|A\|$	$\|R\|$	$\sum \|uv\|$	libsemigroups		ACE		GAP	
					Mean	S.D	Mean	S.D	Mean	S.D
(B.9)	16,384	6	9	34	2.450×10^{-1}	1.2×10^{-3}	3.62×10^{-2}	1.1×10^{-3}	1.99×10^{-1}	2×10^{-3}
(B.10)	8	6	9	34	2.547×10^{-1}	5×10^{-4}	2.14×10^{-2}	8×10^{-4}	1.444×10^{-1}	4×10^{-4}
(B.11)	131,072	6	9	34	1.392×10^{0}	6×10^{-3}	1.88×10^{-1}	5×10^{-3}	1.89×10^{1}	2×10^{-1}
(B.12)	1	6	9	34	1.037×10^{-1}	6×10^{-4}	1.25×10^{-2}	8×10^{-4}	3.08×10^{-2}	2×10^{-4}
(B.13)	131,072	6	9	34	1.80×10^{0}	2×10^{-2}	2.02×10^{-1}	2×10^{-3}	1.880×10^{1}	1.5×10^{-1}
(B.14)	262,144	7	14	50	3.467×10^{0}	4×10^{-3}	6.08×10^{-1}	3×10^{-3}	3.97×10^{1}	3×10^{-1}
(B.15)	29	14	21	49	7.76×10^{-2}	2×10^{-4}	2.44×10^{-2}	1.6×10^{-3}	2.145×10^{-2}	4×10^{-5}
(B.16)	95,040	7	14	60	2.364×10^{-1}	1.0×10^{-3}	5.63×10^{-2}	8×10^{-4}	1.0025×10^{-1}	1.8×10^{-4}
(B.17)	180	4	7	83	4.63×10^{-1}	3×10^{-3}	1.953×10^{-1}	6×10^{-4}	7.66×10^{0}	3×10^{-2}
(B.18)	786,432	9	21	73	1.252×10^{1}	9×10^{-2}	2.305×10^{0}	1.5×10^{-2}	6.752×10^{1}	1.6×10^{-1}

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 16.

$$\begin{aligned}
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 \\
H &= \langle\, bc \,\rangle \qquad (B.9)\\
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 \\
H &= \langle\, bc, a^{-1}b^{-1}(a^{-1})^2bcabc^{-1} \,\rangle \qquad (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 \\
H &= \langle\, ab^{-1}c^{-1}bac, ba^{-1}c^{-1}baaca^{-1}, acca^{-1}a^{-1}b^{-1}ab \,\rangle \qquad (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 \\
H &= \langle\, bc, a^{-1}b^{-1}a^{-1}a^{-1}bcabc^{-1}, a^{-1}c^3acb^{-1}ca^{-1} \,\rangle \qquad (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 \\
H &= \langle\, \varepsilon \,\rangle \qquad (B.13)\\
G &= \langle a, b, c, x \mid ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, \\
&\qquad ac^2a^{-1}a^{-1}b^{-1}ab, x^2, a^{-1}xax, b^{-1}xbx, c^{-1}xcx\rangle \\
H &= \langle\, ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, ac^2a^{-1}a^{-1}b^{-1}ab \,\rangle \qquad (B.14)\\
G &= F_{2,7} = \langle a, b, c, d, x, y, z \mid abc^{-1}, bcd^{-1}, cdx^{-1}, dxy^{-1}, xyz^{-1}, yza^{-1}, zab^{-1}\rangle \\
H &= \langle\, \varepsilon \,\rangle \qquad (B.15)\\
G &= M_{12} = \langle a, b, c \mid a^{11}, b^2, c^2, (ab)^3, (ac)^3, (bc)^{10}, (cb)^2abcbc(a^{-1})^5\rangle \\
H &= \langle\, \varepsilon \,\rangle \qquad (B.16)\\
G &= SL(2, 19) = \langle a, b \mid ab^{-1}a^{-1}b^{-1}a^{-1}b^{-1}, b^{-1}a^{-1}a^{-1}baa, ab^4ab^{10}ab^4ab^{29}a^{12}\rangle \\
H &= \langle\, b \,\rangle \qquad (B.17)\\
G &= \langle a, b, c, x, y \mid x^2, y^3, ab^{-1}c^{-1}bac, ba^{-1}c^{-1}baaca^{-1}, \\
&\qquad acca^{-1}a^{-1}b^{-1}ab, a^{-1}xax, b^{-1}xbx, c^{-1}xcx, \\
&\qquad a^{-1}y^{-1}ay, b^{-1}y^{-1}by, c^{-1}y^{-1}cy, xy^{-1}xy\rangle \\
H &= \langle\, ab^{-1}c^{-1}bac, ba^{-1}c^{-1}ba^2ca^{-1}, ac^2a^{-2}b^{-1}ab \,\rangle \qquad (B.18)
\end{aligned}$$

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. Abram, A., Reutenauer, C.: The stylic monoid. Semigroup Forum **105**, 1–45 (2022). https://doi.org/10.1007/s00233-022-10285-3
2. Arthur, R.E., Ruškuc, N.: Presentations for two extensions of the monoid of order-preserving mappings on a finite chain. Southeast Asian Bull. Math. **24**, 1–7 (2000). https://doi.org/10.1007/s10012-000-0001-1
3. Boone, W.W.: The word problem. Proc. Natl. Acad. Sci. **44**, 1061–1065 (1958). https://doi.org/10.1073/pnas.44.10.1061
4. Cain, A.J., Maltcev, V.: Decision problems for finitely presented and one-relation semigroups and monoids. Int. J. Algebra Comput. **19**, 747–770 (2009). https://doi.org/10.1142/s0218196709005366
5. Cannon, J.J., Dimino, L.A., Havas, G., Watson, J.M.: Implementation and analysis of the Todd–Coxeter algorithm. Math. Comput. **27**, 463–490 (1973). https://doi.org/10.2307/2005654
6. Dehn, M.: Über unendliche diskontinuierliche Gruppen. Math. Ann. **71**, 116–144 (1911). https://doi.org/10.1007/BF01456932
7. Easdown, D., East, J., Fitzgerald, D.G.: A presentation of the dual symmetric inverse monoid. Int. J. Algebra Comput. **18**, 357–374 (2008). https://doi.org/10.1142/s0218196708004470
8. East, J.: Generators and relations for partition monoids and algebras. J. Algebra **339**, 1–26 (2011). https://doi.org/10.1016/j.jalgebra.2011.04.008
9. East, J.: Presentations for Temperley–Lieb algebras. Q. J. Math. **72**, 1253–1269 (2021). https://doi.org/10.1093/qmath/haab001
10. FitzGerald, D.G.: A presentation for the monoid of uniform block permutations. Bull. Aust. Math. Soc. **68**, 317–324 (2003). https://doi.org/10.1017/S0004972700037692
11. Froidure, V., Pin, J.-E.: Algorithms for computing finite semigroups. In: Cucker, F., Shub, M. (eds.) Foundations of Computational Mathematics, pp. 112–126. Springer, Berlin (1997). https://doi.org/10.1007/978-3-642-60539-0_9
12. GAP-Groups: Algorithms, and Programming, Version 4.12.2. The GAP Group (2022). https://www.gap-system.org
13. Gay, J., Hivert, F.: The 0-rook monoid and its representation theory (2019). arXiv:1910.11740 [math.CO]
14. Havas, G., Ramsay, C.: Coset enumeration: ACE version 5.2 (2009). http://staff.itee.uq.edu.au/havas
15. Holt, D.F., Eick, B., O'Brien, E.A.: Handbook of Computational Group Theory. CRC Press, Boca Raton (2004)
16. Howie, J.M.: Fundamentals of Semigroup Theory. London Mathematical Society Monographs. New Series, vol. 12. Oxford University Press, New York (1995)
17. Jura, A.: Coset enumeration in a finitely presented semigroup. Can. Math. Bull. **21**, 37–46 (1978)
18. Leech, J.: Coset enumeration on digital computers. Math. Proc. Camb. Philos. Soc. **59**, 257–267 (1963). https://doi.org/10.1017/s0305004100036872
19. Linton, S.A.: Constructing matrix representations of finitely presented groups. J. Symb. Comput. **12**, 427–438 (1991). https://doi.org/10.1016/S0747-7171(08)80095-8
20. Linton, S.A.: On vector enumeration. Linear Algebra Appl. **192**, 235–248 (1993). https://doi.org/10.1016/0024-3795(93)90245-J

21. Maltcev, V., Mazorchuk, V.: Presentation of the singular part of the Brauer monoid. Math. Bohem. **132**, 297–323 (2007). https://doi.org/10.21136/mb.2007.134125
22. Markov, A.A.: On the impossibility of certain algorithms in the theory of associative systems. Dokl. Akad. Sci. USSR **55**, 587–590 (1947)
23. Mitchell, J.D.: An implementation of the Todd–Coxeter algorithm for semigroups and monoids in Python3 (2021). https://gist.github.com/james-d-mitchell/6b06bc78e2bcdb6dfef53a2654d9f953
24. Mitchell, J.D., et al.: libsemigroups—C++ library for semigroups and monoids, Version 2.7.3 (2024). https://doi.org/10.5281/zenodo.1437752
25. Neubüser, J.: An elementary introduction to coset table methods in computational group theory. In: Campbell, C.M., Robertson, E.F. (eds.) Groups–St Andrews 1981. London Mathematical Society Lecture Notes Series, pp. 1–45. Cambridge University Press, Cambridge (1982). https://doi.org/10.1017/CBO9780511661884.004
26. Neumann, B.H.: Some remarks on semigroup presentations. Can. J. Math. **19**, 1018–1026 (1968)
27. Novikov, P.S.: On the algorithmic unsolvability of the word problem in group theory. Tr. Mat. Inst. Steklova **44**, 3–143 (1955)
28. Robertson, E.F., Ünlü, Y.: On semigroup presentations. Proc. Edinb. Math. Soc. **36**, 55–68 (1992)
29. Rotman, J.J.: An Introduction to the Theory of Groups. Graduate Texts in Mathematics, vol. 148, 4th edn. Springer, New York (1995). https://doi.org/10.1007/978-1-4612-4176-8
30. Ruškuc, N.: Semigroup Presentations. PhD thesis. University of St Andrews (1995)
31. Sims, C.C.: Computation with Finitely Presented Groups. Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge (1994)
32. Stephen, J.B.: Applications of Automata Theory to Presentations of Monoids and Inverse Monoids. PhD thesis. University of Nebraska (1987). https://digitalcommons.unl.edu/dissertations/AAI8803771
33. Stephen, J.: Presentations of inverse monoids. J. Pure Appl. Algebra **63**, 81–112 (1990). https://doi.org/10.1016/0022-4049(90)90057-O
34. Tarjan, R.E.: Efficiency of a good but not linear set union algorithm. J. ACM **22**, 215–225 (1975). https://doi.org/10.1145/321879.321884
35. Todd, J.A., Coxeter, H.S.M.: A practical method for enumerating cosets of a finite abstract group. Proc. Edinb. Math. Soc. **5**, 26–34 (1936). https://doi.org/10.1017/s0013091500008221
36. Walker, T.: Semigroup Enumeration—Computer Implementation and Applications. PhD thesis. University of St Andrews (1992)
37. Walker, T.: tcsemi (1992). https://github.com/james-d-mitchell/tcsemi
38. Williams, A.: Monoid automata factory (2016). http://maffsa.sourceforge.net/