Bounded Automata Groups are co-ET0L

Alex Bishop and Murray Elder(✉)

School of Mathematical and Physical Sciences, University of Technology Sydney, Ultimo, NSW 2007, Australia
{alexander.bishop,murray.elder}@uts.edu.au

Abstract. Holt and Röver proved that finitely generated bounded automata groups have indexed co-word problem. Here we sharpen this result to show they are in fact co-ET0L.

Keywords: Formal language theory · ET0L language · Check-stack pushdown automaton · Bounded automata group · Co-word problem

1 Introduction

A recurrent theme in group theory is to understand and classify group-theoretic problems in terms of their formal language complexity [1,9–11,20]. Many authors have considered the groups whose non-trivial elements, i.e. *co-word problem*, can be described as a context-free language [3,14,16,18]. Holt and Röver went beyond context-free to show that a large class known as *bounded automata groups* have an indexed co-word problem [15]. This class includes important examples such as Grigorchuk's group of intermediate growth, the Gupta-Sidki groups, and many more [12,13,21,24]. For the specific case of the Grigorchuk group, Ciobanu *et al.* [5] showed that the co-word problem was in fact ET0L. ET0L is a class of languages coming from L-systems which lies strictly between context-free and indexed [19,22,23]. Ciobanu *et al.* rely on the grammar description of ET0L for their result. Here we are able to show that all finitely generated bounded automata groups have ET0L co-word problem by instead making use of an equivalent machine description: check-stack pushdown (cspd) automata.

ET0L languages, in particular their deterministic versions, have recently come to prominence in describing solution sets to equations in groups and monoids [4,7,8]. The present paper builds on the recent resurgence of interest, and demonstrates the usefulness of a previously overlooked machine description.

For a group G with finite generating set X, we denote by $\mathrm{coW}(G,X)$ the set of all words in the language $(X\cup X^{-1})^\star$ that represent non-trivial elements in G. We call $\mathrm{coW}(G,X)$ the co-word problem for G (with respect to X). Given a class $\mathcal{C}$ of formal languages that is closed under inverse homomorphism, if $\mathrm{coW}(G,X)$

Research supported by Australian Research Council grant DP160100486 and an Australian Government Research Training Program PhD Scholarship.

C. Martín-Vide et al. (Eds.): LATA 2019, LNCS 11417, pp. 82–94, 2019.
https://doi.org/10.1007/978-3-030-13435-8_6

is in $\mathcal{C}$ then so is $\mathrm{coW}(G, Y)$ for any finite generating set Y of G [14]. Thus, we say that a group is *co-$\mathcal{C}$* if it has a co-word problem in the class $\mathcal{C}$. Note that ET0L is a full AFL [6] and so is closed under inverse homomorphism.

2 ET0L Languages and CSPD Automata

An *alphabet* is a finite set. Let Σ and V be two alphabets which we will call the *terminals* and *non-terminals*, respectively. We will use lower case letters to represent terminals in Σ and upper case letters for non-terminals in V. By $\Sigma^\star$, we will denote the set of words over Σ with $\varepsilon \in \Sigma^\star$ denoting the empty word.

A *table*, τ, is a finite set of context-free replacement rules where each non-terminal, $X \in V$, has at least one replacement in τ. For example, with $\Sigma = \{a, b\}$ and $V = \{S, A, B\}$, the following are tables.

$$\alpha : \begin{cases} S \to SS \mid S \mid AB \\ A \to A \\ B \to B \end{cases} \qquad \beta : \begin{cases} S \to S \\ A \to aA \\ B \to bB \end{cases} \qquad \gamma : \begin{cases} S \to S \\ A \to \varepsilon \\ B \to \varepsilon \end{cases} \tag{1}$$

We apply a table, τ, to the word $w \in (\Sigma \cup V)^\star$ to obtain a word w', written $w \to^\tau w'$, by performing a replacement in τ to each non-terminal in w. If a table includes more than one rule for some non-terminal, we nondeterministically apply any such rule to each occurrence. For example, with $w = SSSS$ and α as in (1), we can apply α to w to obtain $w' = SABSSAB$. Given a sequence of tables $\tau_1, \tau_2, \ldots, \tau_k$, we will write $w \to^{\tau_1\tau_2\cdots\tau_k} w'$ if there is a sequence of words $w = w_1, w_2, \ldots, w_{k+1} = w'$ such that $w_j \to^{\tau_j} w_{j+1}$ for each j.

Definition 1 (Asveld [2]). *An* ET0L grammar *is a 5-tuple* $G = (\Sigma, V, T, \mathcal{R}, S)$*, where*

1. Σ *and* V *are the alphabets of* terminals *and* non-terminals*, respectively;*
2. $T = \{\tau_1, \tau_2, \ldots, \tau_k\}$ *is a finite set of* tables*;*
3. $\mathcal{R} \subseteq T^\star$ *is a regular language called the* rational control*; and*
4. $S \in V$ *is the* start symbol.

The ET0L language *produced by the grammar* G*, denoted* $L(G)$*, is*

$$L(G) := \{w \in \Sigma^\star \,:\, S \to^v w \;\; \textit{for some} \;\; v \in \mathcal{R}\}\,.$$

For example, with α, β and γ as in (1), the language produced by the above grammar with rational control $\mathcal{R} = \alpha^\star\beta^\star\gamma$ is $\{(a^n b^n)^m \,:\, n, m \in \mathbb{N}\}$.

2.1 CSPD Automata

A *cspd automaton*, introduced in [17], is a nondeterministic finite-state automaton with a one-way input tape, and access to both a *check-stack* (with stack alphabet Δ) and a pushdown stack (with stack alphabet Γ), where access to

these two stacks is tied in a very particular way. The execution of a cspd machine can be separated into two stages.

In the first stage the machine is allowed to push to its check-stack but *not* its pushdown, and further, the machine will not be allowed to read from its input tape. Thus, the set of all possible check-stacks that can be constructed in this stage forms a regular language which we will denote as $\mathcal{R}$.

In the second stage, the machine can no longer alter its check-stack, but is allowed to access its pushdown and input tape. We restrict the machine's access to its stacks so that it can only move along its check-stack by pushing and popping items to and from its pushdown. In particular, every time the machine pushes a value onto the pushdown it will move up the check-stack, and every time it pops a value off of the pushdown it will move down the check-stack; see Fig. 1 for an example of this behaviour.

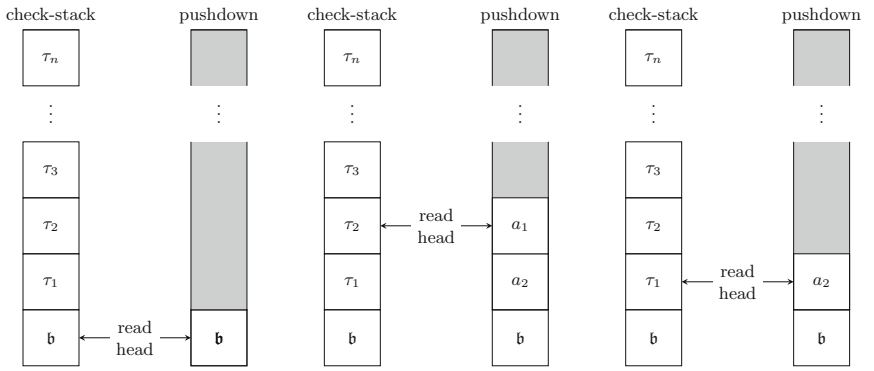

Fig. 1. An example of a cspd machine pushing $w = a_1a_2$, where $a_1, a_2 \in \Delta$, onto its pushdown stack, then popping a_1

We define a cspd machine formally as follows.

Definition 2. *A* cspd machine *is a 9-tuple* $\mathcal{M} = (Q, \Sigma, \Gamma, \Delta, \mathfrak{b}, \mathcal{R}, \theta, q_0, F)$, *where*

1. Q *is the set of* states*;*
2. Σ *is the* input alphabet*;*
3. Γ *is the* alphabet for the pushdown*;*
4. Δ *is the* alphabet for the check-stack*;*
5. $\mathfrak{b} \notin \Delta \cup \Gamma$ *is the* bottom of stack symbol*;*
6. $\mathcal{R} \subseteq \Delta^\star$ *is a* regular language of allowed check-stack contents*;*
7. θ *is a finite subset of*

$$(Q \times (\Sigma \cup \{\varepsilon\}) \times ((\Delta \times \Gamma) \cup \{(\varepsilon, \varepsilon), (\mathfrak{b}, \mathfrak{b})\})) \times (Q \times (\Gamma \cup \{\mathfrak{b}\})^\star),$$

and is called the transition relation *(see below for allowable elements of* θ*);*

8. $q_0 \in Q$ *is the* start state*; and*
9. $F \subseteq Q$ *is the set of* accepting states.

In its initial configuration, the machine will be in state q_0, the check-stack will contain $\mathfrak{b}w$ for some nondeterministic choice of $w \in \mathcal{R}$, the pushdown will contain only the letter $\mathfrak{b}$, the read-head for the input tape will be at its first letter, and the read-head for the machine's stacks will be pointing to the $\mathfrak{b}$ on both stacks. From here, the machine will follow transitions as specified by θ, each such transition having one of the following three forms, where $a \in \Sigma \cup \{\varepsilon\}$, $p, q \in Q$ and $w \in \Gamma^\star$.

1. $((p, a, (\mathfrak{b}, \mathfrak{b})), (q, w\mathfrak{b})) \in \theta$ meaning that if the machine is in state p, sees $\mathfrak{b}$ on both stacks and is able to consume a from its input; then it can follow this transition to consume a, push w onto the pushdown and move to state q.
2. $((p, a, (d, g)), (q, w)) \in \theta$ where $(d, g) \in \Delta \times \Gamma$, meaning that if the machine is in state p, sees d on its check-stack, g on its pushdown, and is able to consume a from its input; then it can follow this transition to consume a, pop g, then push w and move to state q.
3. $((p, a, (\varepsilon, \varepsilon)), (q, w)) \in \theta$ meaning that if the machine is in state p and can consume a from its input; then it can follow this transition to consume a, push w and move to state q.

In the previous three cases, $a = \varepsilon$ corresponds to a transition in which the machine does not consume a letter from input. We use the convention that, if $w = w_1 w_2 \cdots w_k$ with each $w_j \in \Gamma$, then the machine will first push the w_k, followed by the w_{k-1} and so forth. The machine accepts if it has consumed all its input and is in an accepting state $q \in F$.

In [17] van Leeuwen proved that the class of languages that are recognisable by cspd automata is precisely the class of ET0L languages.

3 Bounded Automata Groups

For $d \geqslant 2$, let $\mathcal{T}_d$ denote the d-regular rooted tree, that is, the infinite rooted tree where each vertex has exactly d children. We identify the vertices of $\mathcal{T}_d$ with words in $\Sigma^\star$ where $\Sigma = \{a_1, a_2, \ldots, a_d\}$. In particular, we will identify the root with the empty word $\varepsilon \in \Sigma^\star$ and, for each vertex $v \in \mathrm{V}(\mathcal{T}_d)$, we will identify the k-th child of v with the word va_k, see Fig. 2.

Recall that an automorphism of a graph is a bijective mapping of the vertex set that preserves adjacencies. Thus, an automorphism of $\mathcal{T}_d$ preserves the root and "levels" of the tree. The set of all automorphisms of $\mathcal{T}_d$ is a group, which we denote by $\mathrm{Aut}(\mathcal{T}_d)$. We denote the *permutation group of* Σ as $\mathrm{Sym}(\Sigma)$. An important observation is that $\mathrm{Aut}(\mathcal{T}_d)$ can be seen as the wreath product $\mathrm{Aut}(\mathcal{T}_d) \wr \mathrm{Sym}(\Sigma)$, since any automorphism $\alpha \in \mathrm{Aut}(\mathcal{T}_d)$ can be written uniquely as $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d) \cdot \sigma$ where $\alpha_i \in \mathrm{Aut}(\mathcal{T}_d)$ is an automorphism of the sub-tree with root a_i, and $\sigma \in \mathrm{Sym}(\Sigma)$ is a permutation of the first level. Let $\alpha \in \mathrm{Aut}(\mathcal{T}_d)$ where $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d) \cdot \sigma \in \mathrm{Aut}(\mathcal{T}_d) \wr \mathrm{Sym}(\Sigma)$. For any $b = a_i \in \Sigma$, the *restriction of* α *to* b, denoted $\alpha|_b := \alpha_i$, is the action of α on the sub-tree

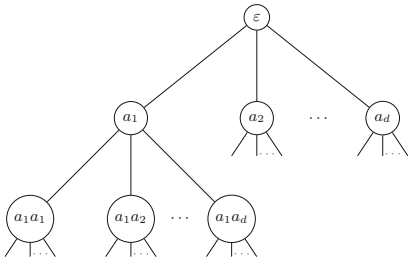

Fig. 2. A labelling of the vertices of $\mathcal{T}_d$ with the root labelled ε

with root b. Given any vertex $w = w_1w_2 \cdots w_k \in \Sigma^\star$ of $\mathcal{T}_d$, we can define the *restriction of* α *to* w recursively as

$$\alpha|_w = \left(\alpha|_{w_1w_2\cdots w_{k-1}}\right)\Big|_{w_k}$$

and thus describe the action of α on the sub-tree with root w.

A *Σ-automaton*, (Γ, v), is a finite directed graph with a distinguished vertex v, called the *initial state*, and a $(\Sigma \times \Sigma)$-labelling of its edges, such that each vertex has exactly $|\Sigma|$ outgoing edges: with one outgoing edge with a label of the form $(a, \cdot)$ and one with a label of the form $(\cdot, a)$ for each $a \in \Sigma$.

Given some Σ-automaton (Γ, v), where $\Sigma = \{a_1, \ldots, a_d\}$, we can define an automorphism $\alpha_{(\Gamma,v)} \in \mathrm{Aut}(\mathcal{T}_d)$ as follows. For any given vertex $b_1b_2 \cdots b_k \in \Sigma^\star = \mathrm{V}(\mathcal{T}_d)$, there exists a unique path in Γ starting from the initial vertex, v, of the form $(b_1, b_1')\,(b_2, b_2') \cdots (b_k, b_k')$, thus we will now define $\alpha_{(\Gamma,v)}$ such that $\alpha_{(\Gamma,v)}(b_1b_2 \cdots b_k) = b_1'b_2' \cdots b_k'$. Notice that it follows from the definition of a Σ-automaton that $\alpha_{(\Gamma,v)}$ is a tree automorphism as required.

An *automaton automorphism*, α, of the tree $\mathcal{T}_d$ is an automorphism for which there exists a Σ-automaton, (Γ, v), such that $\alpha = \alpha_{(\Gamma,v)}$. The set of all automaton automorphisms of the tree $\mathcal{T}_d$ form a group which we will denote as $\mathcal{A}(\mathcal{T}_d)$. A subgroup of $\mathcal{A}(\mathcal{T}_d)$ is called an *automata group*.

An automorphism $\alpha \in \mathrm{Aut}(\mathcal{T}_d)$ will be called *bounded* (originally defined in [24]) if there exists a constant $N \in \mathbb{N}$ such that for each $k \in \mathbb{N}$, there are no more than N vertices $v \in \Sigma^\star$ with $|v| = k$ (i.e. at level k) such that $\alpha|_v \neq 1$. Thus, the action of such a bounded automorphism will, on each level, be trivial on all but (up to) N sub-trees. The set of all such automorphisms form a group which we will denote as $\mathcal{B}(\mathcal{T}_d)$. The group of all *bounded automaton automorphisms* is defined as the intersection $\mathcal{A}(\mathcal{T}_d) \cap \mathcal{B}(\mathcal{T}_d)$, which we will denote as $\mathcal{D}(\mathcal{T}_d)$. A subgroup of $\mathcal{D}(\mathcal{T}_d)$ is called a *bounded automata group*.

A *finitary automorphism* of $\mathcal{T}_d$ is an automorphism ϕ such that there exists a constant $k \in \mathbb{N}$ for which $\phi|_v = 1$ for each $v \in \Sigma^\star$ with $|v| = k$. Thus, a finitary automorphism is one that is trivial after some k levels of the tree. Given a finitary automorphism ϕ, the smallest k for which this definition holds will be called its *depth* and will be denoted as $\mathrm{depth}(\phi)$. We will denote the group formed by all finitary automorphisms of $\mathcal{T}_d$ as $\mathrm{Fin}(\mathcal{T}_d)$. See Fig. 3 for examples

of the actions of finitary automorphisms on their associated trees (where any unspecified sub-tree is fixed by the action).

Fig. 3. Examples of finitary automorphisms $a, b \in \mathrm{Fin}(\mathcal{T}_2)$

Let $\delta \in \mathcal{A}(\mathcal{T}_d) \setminus \mathrm{Fin}(\mathcal{T}_d)$. We call δ a *directed automaton automorphism* if

$$\delta = (\phi_1, \phi_2, \ldots, \phi_{k-1}, \delta', \phi_{k+1}, \ldots, \phi_d) \cdot \sigma \in \mathrm{Aut}(\mathcal{T}_d) \wr \mathrm{Sym}(\Sigma) \tag{2}$$

where each ϕ_j is finitary and δ' is also directed automaton (that is, not finitary and can also be written in this form). We call $\mathrm{dir}(\delta) = b = a_k \in \Sigma$, where $\delta' = \delta|_b$ is directed automaton, the *direction* of δ; and we define the *spine* of δ, denoted $\mathrm{spine}(\delta) \in \Sigma^\omega$, recursively such that $\mathrm{spine}(\delta) = \mathrm{dir}(\delta)\,\mathrm{spine}(\delta')$. We will denote the set of all directed automaton automorphisms as $\mathrm{Dir}(\mathcal{T}_d)$. See Fig. 4 for examples of directed automaton automorphisms (in which a and b are the finitary automorphisms in Fig. 3).

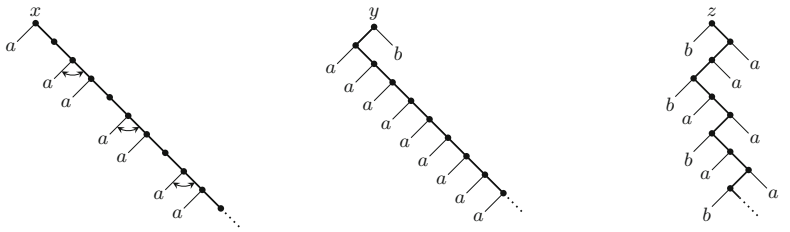

Fig. 4. Examples of directed automata automorphisms $x, y, z \in \mathrm{Dir}(\mathcal{T}_2)$

The following lemma is essential to prove our main theorem.

Lemma 3. *The spine,* $\mathrm{spine}(\delta) \in \Sigma^\omega$*, of a directed automaton automorphism,* $\delta \in \mathrm{Dir}(\mathcal{T}_d)$*, is eventually periodic, that is, there exists some* $\iota = \iota_1\iota_2\cdots\iota_s \in \Sigma^\star$*, called the* initial section*, and* $\pi = \pi_1\pi_2\cdots\pi_t \in \Sigma^\star$*, called the* periodic section*, such that* $\mathrm{spine}(\delta) = \iota\,\pi^\omega$*; and*

$$\delta|_{\iota\,\pi^k\,\pi_1\pi_2\cdots\pi_j} = \delta|_{\iota\,\pi_1\pi_2\cdots\pi_j} \tag{3}$$

for each $k, j \in \mathbb{N}$ *with* $0 \leqslant j < t$.

Proof. Let (Γ, v) be a Σ-automaton such that $\delta = \alpha_{(\Gamma,v)}$. By the definition of Σ-automata, for any given vertex $w = w_1 w_2 \cdots w_k \in \Sigma^\star$ of $\mathcal{T}_d$ there exists a vertex $v_w \in \mathrm{V}(\Gamma)$ such that $\delta|_w = \alpha_{(\Gamma,v_w)}$. In particular, such a vertex v_w can be obtained by following the path with edges labelled $(w_1, w_1')(w_2, w_2') \cdots (w_k, w_k')$. Then, since there are only finitely many vertices in Γ, the set of all restrictions of δ is finite, that is,

$$\left|\left\{ \delta|_w = \alpha_{(\Gamma,v_w)} \,:\, w \in \Sigma^\star \right\}\right| < \infty. \tag{4}$$

Let $b = b_1 b_2 b_3 \cdots = \mathrm{spine}(\delta) \in \Sigma^\omega$ denote the spine of δ. Then, there exists some $n, m \in \mathbb{N}$ with $n < m$ such that

$$\delta|_{b_1 b_2 \cdots b_n} = \delta|_{b_1 b_2 \cdots b_n \cdots b_m} \tag{5}$$

as otherwise there would be infinitely many distinct restrictions of the form $\delta|_{b_1 b_2 \cdots b_k}$ thus contradicting (4). By the definition spine, it follows that

$$\mathrm{spine}\left(\delta|_{b_1 b_2 \cdots b_n}\right) = (b_{n+1} b_{n+2} \cdots b_m)\ \mathrm{spine}\left(\delta|_{b_1 b_2 \cdots b_n \cdots b_m}\right).$$

and hence, by (5),

$$\mathrm{spine}\left(\delta|_{b_1 b_2 \cdots b_n}\right) = (b_{n+1} b_{n+2} \cdots b_m)^\omega.$$

Thus,

$$\mathrm{spine}(\delta) = (b_1 b_2 \cdots b_n)\ \mathrm{spine}\left(\delta|_{b_1 b_2 \cdots b_n}\right) = (b_1 b_2 \cdots b_n)\ (b_{n+1} b_{n+2} \cdots b_m)^\omega.$$

Then, by taking $\iota = b_1 b_2 \cdots b_n$ and $\pi = b_{n+1} b_{n+2} \cdots b_m$, we have $\mathrm{spine}(\delta) = \iota\, \pi^\omega$. Moreover, from (5), we have Eq. (3) as required. □

Notice that each finitary and directed automata automorphism is also bounded, in fact, we have the following proposition which shows that the generators of any given bounded automata group can be written as words in $\mathrm{Fin}(\mathcal{T}_d)$ and $\mathrm{Dir}(\mathcal{T}_d)$.

Proposition 4 (Proposition 16 in [24]). *The group $\mathcal{D}(\mathcal{T}_d)$ of bounded automata automorphisms is generated by $\mathrm{Fin}(\mathcal{T}_d)$ together with $\mathrm{Dir}(\mathcal{T}_d)$.*

4 Main Theorem

Theorem 5. *Every finitely generated bounded automata group is co-ET0L.*

The idea of the proof is straightforward: we construct a cspd machine that chooses a vertex $v \in \mathrm{V}(\mathcal{T}_d)$, writing its label on the check-stack and a copy on its pushdown; as it reads letters from input, it uses the pushdown to keep track of where the chosen vertex is moved; and finally it checks that the pushdown and check-stack differ. The full details are as follows.

Proof. Let $G \leqslant \mathcal{D}(\mathcal{T}_d)$ be a bounded automata group with finite symmetric generating set X. By Proposition 4, we can define a map

$$\varphi : X \to (\mathrm{Fin}(\mathcal{T}_d) \cup \mathrm{Dir}(\mathcal{T}_d))^\star$$

so that $x =_{\mathcal{D}(\mathcal{T}_d)} \varphi(x)$ for each $x \in X$. Let

$$Y = \left\{\alpha \in \mathrm{Fin}(\mathcal{T}_d) \cup \mathrm{Dir}(\mathcal{T}_d) \,:\, \alpha \text{ or } \alpha^{-1} \text{ is a factor of } \varphi(x) \text{ for some } x \in X\right\}$$

which is finite and symmetric. Consider the group $H \leqslant \mathcal{D}(\mathcal{T}_d)$ generated by Y. Since ET0L is closed under inverse word homomorphism, it suffices to prove that $\mathrm{coW}(H, Y)$ is ET0L, as $\mathrm{coW}(G, X)$ is its inverse image under the mapping $X^\star \to Y^\star$ induced by φ. We construct a cspd machine $\mathcal{M}$ that recognises $\mathrm{coW}(H, Y)$, thus proving that G is co-ET0L.

Let $\alpha = \alpha_1\alpha_2\cdots\alpha_n \in Y^\star$ denote an input word given to $\mathcal{M}$. The execution of the cspd will be separated into four stages; (1) choosing a vertex $v \in \Sigma^\star$ of $\mathcal{T}_d$ which witnesses the non-triviality of α (and placing it on the stacks); (2a) reading a finitary automorphism from the input tape; (2b) reading a directed automaton automorphism from the input tape; and (3) checking that the action of α on v that it has computed is non-trivial.

After Stage 1, $\mathcal{M}$ will be in state q_{comp}. From here, $\mathcal{M}$ nondeterministically decides to either read from its input tape, performing either Stage 2a or 2b and returning to state q_{comp}; or to finish reading from input by performing Stage 3.

We set both the check-stack and pushdown alphabets to be $\Delta = \Gamma = \Sigma \sqcup \{\mathfrak{t}\}$.

Stage 1: Choosing a Witness $v = v_1v_2\cdots v_m \in \Sigma^\star$.

If α is non-trivial, then there must exist a vertex $v \in \Sigma^\star$ such that $\alpha \cdot v \neq v$. Thus, we nondeterministically choose such a witness from $\mathcal{R} = \Sigma^\star\mathfrak{t}$ and store it on the check-stack, where the letter $\mathfrak{t}$ represents the top of the check-stack.

From the start state, q_0, $\mathcal{M}$ will copy the contents of the check-stack onto the pushdown, then enter the state $q_{\mathrm{comp}} \in Q$. Formally, this will be achieved by adding the transitions (for each $a \in \Sigma$):

$$((q_0, \varepsilon, (\mathfrak{b}, \mathfrak{b})), (q_0, \mathfrak{t}\mathfrak{b})),\ ((q_0, \varepsilon, (a, \mathfrak{t})), (q_0, \mathfrak{t}a)),\ ((q_0, \varepsilon, (\mathfrak{t}, \mathfrak{t})), (q_{\mathrm{comp}}, \mathfrak{t})).$$

This stage concludes with $\mathcal{M}$ in state q_{comp}, and the read-head pointing to $(\mathfrak{t}, \mathfrak{t})$. Note that whenever the machine is in state q_{comp} and $\alpha_1\alpha_2\cdots\alpha_k$ has been read from input, then the contents of pushdown will represent the permuted vertex $(\alpha_1\alpha_2\cdots\alpha_k)\cdot v$. Thus, the two stacks are initially the same as no input has been read and thus no group action has been simulated. In Stages 2a and 2b, only the height of the check-stack is impotant, that is, the exact contents of the check-stack will become relevant in Stage 3.

Stage 2a: Reading a Finitary Automorphism $\phi \in Y \cap \mathrm{Fin}(\mathcal{T}_d)$.

By definition, there exists some $k_\phi = \mathrm{depth}(\phi) \in \mathbb{N}$ such that $\phi|_u = 1$ for each $u \in \Sigma^\star$ for which $|u| \geqslant k_\phi$. Thus, given a vertex $v = v_1v_2\cdots v_m \in \Sigma^\star$, we have

$$\phi(v) = \phi(v_1v_2\cdots v_{k_\phi})\ v_{(k_\phi+1)}\cdots v_m.$$

Given that $\mathcal{M}$ is in state q_{comp} with $\mathfrak{t}v_1v_2\cdots v_m\mathfrak{b}$ on its pushdown, we will read ϕ from input, move to state $q_{\phi,\varepsilon}$ and pop the $\mathfrak{t}$; we will then pop the next k_ϕ (or fewer if $m < k_\phi$) letters off the pushdown, and as we are popping these letters we visit the sequence of states q_{ϕ,v_1}, q_{ϕ,v_1v_2}, $\ldots$, $q_{\phi,v_1v_2\cdots v_{k_\phi}}$. From the final state in this sequence, we then push $\mathfrak{t}\phi(v_1\cdots v_{k_\phi})$ onto the pushdown, and return to the state q_{comp}.

Formally, for letters $a, b \in \Sigma$, $\phi \in Y \cap \text{Fin}(\mathcal{T}_d)$, and vertices $u, w \in \Sigma^\star$ where $|u| < k_\phi$ and $|w| = k_\phi$, we have the transitions

$$((q_{\text{comp}}, \phi, (\mathfrak{t}, \mathfrak{t})), (q_{\phi,\varepsilon}, \varepsilon)),\ ((q_{\phi,u}, \varepsilon, (a, b)), (q_{\phi,ub}, \varepsilon)),$$

$$((q_{\phi,w}, \varepsilon, (\varepsilon, \varepsilon)), (q_{\text{comp}}, \mathfrak{t}\phi(w)))$$

for the case where $m > k_\phi$, and

$$((q_{\phi,u}, \varepsilon, (\mathfrak{b}, \mathfrak{b})), (q_{\text{comp}}, \mathfrak{t}\phi(u)\mathfrak{b}))$$

for the case where $m \leqslant k_\phi$. Notice that we have finitely many states and transitions since Y, Σ and each k_ϕ is finite.

Stage 2b: Reading a Directed Automorphism $\delta \in Y \cap \text{Dir}(\mathcal{T}_d)$.

By Lemma 3, there exists some $\iota = \iota_1\iota_2\cdots\iota_s \in \Sigma^\star$ and $\pi = \pi_1\pi_2\cdots\pi_t \in \Sigma^\star$ such that $\text{spine}(\delta) = \iota\,\pi^\omega$ and

$$\delta(\iota\pi^\omega) = I_1I_2\cdots I_s\ (\Pi_1\Pi_2\cdots\Pi_t)^\omega$$

where

$$I_i = \delta|_{\iota_1\iota_2\cdots\iota_{i-1}}(\iota_i) \qquad \text{and} \qquad \Pi_j = \delta|_{\iota\,\pi_1\pi_2\cdots\pi_{j-1}}(\pi_j).$$

Given some vertex $v = v_1v_2\cdots v_m \in \Sigma^\star$, let $\ell \in \mathbb{N}$ be largest such that $p = v_1v_2\cdots v_\ell$ is a prefix of the sequence $\iota\pi^\omega = \text{spine}(\delta)$. Then by definition of directed automorphism, $\delta' = \delta|_p$ is directed and $\phi = \delta|_a$, where $a = v_\ell$, is finitary. Then, either $p = \iota_1\iota_2\cdots\iota_\ell$ and

$$\delta(u) = (I_1I_2\cdots I_\ell)\ \delta'(a)\ \phi(v_{\ell+2}v_{\ell+3}\cdots v_m),$$

or $p = \iota\pi^k\pi_1\pi_2\cdots\pi_j$, with $\ell = |\iota| + k\cdot|\pi| + j$, and

$$\delta(u) = (I_1I_2\cdots I_s)\,(\Pi_1\Pi_2\cdots\Pi_t)^k\,(\Pi_1\Pi_2\cdots\Pi_j)\ \delta'(a)\ \phi(v_{\ell+2}v_{\ell+3}\cdots v_m).$$

Hence, from state q_{comp} with $\mathfrak{t}v_1v_2\cdots v_m\mathfrak{b}$ on its pushdown, $\mathcal{M}$ reads δ from input, moves to state $q_{\delta,\iota,0}$ and pops the $\mathfrak{t}$; it then pops pa off the pushdown, using states to remember the letter a and the part of the prefix to which the final letter of p belongs (i.e. ι_i or π_j). From here, $\mathcal{M}$ performs the finitary automorphism ϕ on the remainder of the pushdown (using the same construction as Stage 2a), then, in a sequence of transitions, pushes $\mathfrak{t}\delta(p)\delta'(a)$ and returns to state q_{comp}. The key idea here is that, using only the knowledge of the letter a, the part of ι or π to which the final letter of p belongs, and the height of the check-stack, that $\mathcal{M}$ is able to recover $\delta(p)\delta'(a)$.

We now give the details of the states and transitions involved in this stage of the construction.

We have states $q_{\delta,\iota,i}$ and $q_{\delta,\pi,j}$ with $0 \leqslant i \leqslant |\iota|$, $1 \leqslant j \leqslant |\pi|$; where $q_{\delta,\iota,i}$ represents that the word $\iota_1\iota_2\cdots\iota_i$ has been popped off the pushdown, and $q_{\delta,\pi,j}$ represents that a word $\iota\pi^k\pi_1\pi_2\cdots\pi_j$ for some $k \in \mathbb{N}$ has been popped of the pushdown. Thus, we begin with the transition

$$((q_{\text{comp}}, \delta, (\mathfrak{t}, \mathfrak{t})), (q_{\delta,\iota,0}, \varepsilon)),$$

then for each $i, j \in \mathbb{N}$, $a \in \Sigma$ with $0 \leqslant i < |\iota|$ and $1 \leqslant j < |\pi|$, we have transitions

$$((q_{\delta,\iota,i}, \varepsilon, (a, \iota_{i+1})), (q_{\delta,\iota,(i+1)}, \varepsilon)), \quad ((q_{\delta,\iota,|\iota|}, \varepsilon, (a, \pi_1)), (q_{\delta,\pi,1}, \varepsilon)),$$
$$((q_{\delta,\pi,j}, \varepsilon, (a, \pi_{j+1})), (q_{\delta,\pi,(j+1)}, \varepsilon)), \quad ((q_{\delta,\pi,|\pi|}, \varepsilon, (a, \pi_1)), (q_{\delta,\pi,1}, \varepsilon))$$

to consume the prefix p.

After this, $\mathcal{M}$ will either be at the bottom of its stacks, or its read-head will see a letter on the pushdown that is not the next letter in the spine of δ. Thus, for each $i, j \in \mathbb{N}$ with $0 \leqslant i \leqslant |\iota|$ and $1 \leqslant j \leqslant |\pi|$ we have states $q_{\delta,\iota,i,a}$ and $q_{\delta,\pi,j,a}$; and for each $b \in \Sigma$ we have transitions

$$((q_{\delta,\iota,i}, \varepsilon, (b, a)), (q_{\delta,\iota,i,a}, \varepsilon))$$

where $a \neq \iota_{i+1}$ when $i < |\iota|$ and $a \neq \pi_1$ otherwise, and

$$((q_{\delta,\pi,j}, \varepsilon, (b, a)), (q_{\delta,\pi,j,a}, \varepsilon))$$

where $a \neq \pi_{j+1}$ when $j < |\pi|$ and $a \neq \pi_1$ otherwise.

Hence, after these transitions, $\mathcal{M}$ has consumed pa from its pushdown and will either be at the bottom of its stacks in some state $q_{\delta,\iota,i}$ or $q_{\delta,\pi,j}$; or will be in some state $q_{\delta,\iota,i,a}$ or $q_{\delta,\pi,j,a}$. Note here that, if $\mathcal{M}$ is in the state $q_{\delta,\iota,i,a}$ or $q_{\delta,\pi,j,a}$, then from Lemma 3 we know $\delta' = \delta|_p$ is equivalent to $\delta|_{\iota_1\iota_2\cdots\iota_i}$ or $\delta|_{\iota\pi_1\pi_2\cdots\pi_j}$, respectively; and further, we know the finitary automorphism $\phi = \delta|_{pa} = \delta'|_a$.

Thus, for each state $q_{\delta,\iota,i,a}$ and $q_{\delta,\pi,a}$ we will follow a similar construction to Stage 2a, to perform the finitary automorphism ϕ to the remaining letters on the pushdown, then push $\delta'(a)$ and return to the state $r_{\delta,\iota,i}$ or $r_{\delta,\pi,j}$, respectively. For the case where $\mathcal{M}$ is at the bottom of its stacks we have transitions

$$((q_{\delta,\iota,i}, \varepsilon, (\mathfrak{b}, \mathfrak{b})), (r_{\delta,\iota,i}, \mathfrak{b})), \quad ((q_{\delta,\pi,i}, \varepsilon, (\mathfrak{b}, \mathfrak{b})), (r_{\delta,\pi,i}, \mathfrak{b}))$$

with $0 \leqslant i \leqslant |\iota|$, $1 \leqslant j \leqslant |\pi|$.

Thus, after following these transitions, $\mathcal{M}$ is in some state $r_{\delta,\iota,i}$ or $r_{\delta,\pi,j}$ and all that remains is for $\mathcal{M}$ to push $\delta(p)$ with $p = \iota_1\iota_2\cdots\iota_i$ or $p = \iota\pi^k\pi_1\pi_2\cdots\pi_k$, respectively, onto its pushdown. Thus, for each $i, j \in \mathbb{N}$ with $0 \leqslant i \leqslant |\iota|$ and $1 \leqslant j \leqslant |\pi|$, we have transitions

$$((r_{\delta,\pi,i}, \varepsilon, (\varepsilon, \varepsilon)), (q_{\text{comp}}, \mathfrak{t}I_1I_2\cdots I_i)), \quad ((r_{\delta,\pi,j}, \varepsilon, (\varepsilon, \varepsilon)), (r_{\delta,\pi}, \Pi_1\Pi_2\cdots\Pi_j))$$

where from the state $r_{\delta,\pi}$, through a sequence of transitions, $\mathcal{M}$ will push the remaining $I\Pi^k$ onto the pushdown. In particular, we have transitions

$$((r_{\delta,\pi}, \varepsilon, (\varepsilon, \varepsilon)), (r_{\delta,\pi}, \Pi)), \quad ((r_{\delta,\pi}, \varepsilon, (\varepsilon, \varepsilon)), (q_{\text{comp}}, \mathfrak{t}I)),$$

so that $\mathcal{M}$ can nondeterministically push some number of Π's followed by $\mathfrak{t}I$ before it finishes this stage of the computation. We can assume that the machine pushes the correct number of Π's onto its pushdown as otherwise it will not see $\mathfrak{t}$ on its check-stack while in state q_{comp} and thus would not be able to continue with its computation, as every subsequent stage (2a, 2b, 3) of the computation begins with the read-head pointing to $\mathfrak{t}$ on both stacks.

Once again it is clear that this stage of the construction requires only finitely many states and transitions.

Stage 3: Checking that the Action is Non-trivial.

At the beginning of this stage, the contents of the check-stack represent the chosen witness, v, and the contents of the pushdown represent the action of the input word, α, on the witness, i.e., $\alpha \cdot v$.

In this stage $\mathcal{M}$ checks if the contents of its check-stack and pushdown differ. Formally, we have states q_{accept} and q_{check}, with q_{accept} accepting; for each $a \in \Sigma$, we have transitions

$$((q_{\text{comp}}, \varepsilon, (\mathfrak{t}, \mathfrak{t})), (q_{\text{check}}, \varepsilon)), \quad ((q_{\text{check}}, \varepsilon, (a, a)), (q_{\text{check}}, \varepsilon))$$

to pop identical entries of the pushdown; and for each $(a, b) \in \Sigma \times \Sigma$ with $a \neq b$ we have a transition

$$((q_{\text{check}}, \varepsilon, (a, b)), (q_{\text{accept}}, \varepsilon))$$

to accept if the stacks differ by a letter.

Observe that if the two stacks are identical, then there is no path to the accepting state, q_{accept}, and thus $\mathcal{M}$ will reject. Notice also that by definition of cspd automata, if $\mathcal{M}$ moves into q_{check} before all input has been read, then $\mathcal{M}$ will not accept, i.e., an accepting state is only effective if all input is consumed.

Soundness and Completeness.

If α is non-trivial, then there is a vertex $v \in \Sigma^\star$ such that $\alpha \cdot v \neq v$, which $\mathcal{M}$ can nondeterministically choose to write on its check-stack and thus accept α. If α is trivial, then $\alpha \cdot v = v$ for each vertex $v \in \Sigma^\star$, and there is no choice of checking stack for which $\mathcal{M}$ will accept, so $\mathcal{M}$ will reject.

Thus, $\mathcal{M}$ accepts a word if and only if it is in $\text{coW}(H, Y)$. Hence, the co-word problem $\text{coW}(H, Y)$ is ET0L, completing our proof. □

Acknowledgments. The authors wish to thank Claas Röver, Michal Ferov and Laura Ciobanu for helpful comments.

References

1. Anīsīmov, A.V.: The group languages. Kibernetika (Kiev) **7**(4), 18–24 (1971)
2. Asveld, P.R.J.: Controlled iteration grammars and full hyper-AFL's. Inf. Control **34**(3), 248–269 (1977)
3. Bleak, C., Matucci, F., Neunhöffer, M.: Embeddings into Thompson's group V and $co\mathcal{CF}$ groups. J. Lond. Math. Soc. (2) **94**(2), 583–597 (2016). https://doi.org/10.1112/jlms/jdw04

4. Ciobanu, L., Diekert, V., Elder, M.: Solution sets for equations over free groups are EDT0L languages. Int. J. Algebra Comput. **26**(5), 843–886 (2016). https://doi.org/10.1142/S0218196716500363
5. Ciobanu, L., Elder, M., Ferov, M.: Applications of L systems to group theory. Int. J. Algebra Comput. **28**(2), 309–329 (2018). https://doi.org/10.1142/S0218196718500145
6. Culik II, K.: On some families of languages related to developmental systems. Int. J. Comput. Math. **4**, 31–42 (1974). https://doi.org/10.1080/00207167408803079
7. Diekert, V., Elder, M.: Solutions of twisted word equations, EDT0L languages, and context-free groups. In: 44th International Colloquium on Automata, Languages, and Programming, LIPIcs. Leibniz International Proceedings in Informatics, vol. 80, Article No. 96, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern (2017)
8. Diekert, V., Jeż, A., Kufleitner, M.: Solutions of word equations over partially commutative structures. In: 43rd International Colloquium on Automata, Languages, and Programming, LIPIcs. Leibniz International Proceedings in Informatics, vol. 55, Article No. 127, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern (2016)
9. Elder, M., Kambites, M., Ostheimer, G.: On groups and counter automata. Int. J. Algebra Comput. **18**(8), 1345–1364 (2008). https://doi.org/10.1142/S0218196708004901
10. Epstein, D.B.A., Cannon, J.W., Holt, D.F., Levy, S.V.F., Paterson, M.S., Thurston, W.P.: Word Processing in Groups. Jones and Bartlett Publishers, Boston (1992)
11. Gilman, R.H.: Formal languages and infinite groups. In: Geometric and Computational Perspectives on Infinite Groups, Minneapolis, MN and New Brunswick, NJ, 1994. DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, vol. 25, pp. 27–51. American Mathematical Society, Providence (1996)
12. Grigorchuk, R.: On Burnside's problem on periodic groups. Funktsional. Anal. i Prilozhen. **14**(1), 53–54 (1980)
13. Gupta, N., Sidki, S.: On the Burnside problem for periodic groups. Mathematische Zeitschrift **182**(3), 385–388 (1983). https://doi.org/10.1007/BF01179757
14. Holt, D.F., Rees, S., Röver, C.E., Thomas, R.M.: Groups with context-free co-word problem. J. Lond. Math. Soc. (2) **71**(3), 643–657 (2005). https://doi.org/10.1112/S002461070500654X
15. Holt, D.F., Röver, C.E.: Groups with indexed co-word problem. Int. J. Algebra Comput. **16**(5), 985–1014 (2006). https://doi.org/10.1142/S0218196706003359
16. König, D., Lohrey, M., Zetzsche, G.: Knapsack and subset sum problems in nilpotent, polycyclic, and co-context-free groups. In: Algebra and Computer Science, Contemporary Mathematics, vol. 677, pp. 129–144. American Mathematical Society, Providence (2016)
17. van Leeuwen, J.: Variations of a new machine model. In: 17th Annual Symposium on Foundations of Computer Science, Houston, Texas 1976, pp. 228–235. IEEE Computer Society, Long Beach, October 1976. https://doi.org/10.1109/SFCS.1976.35
18. Lehnert, J., Schweitzer, P.: The co-word problem for the Higman-Thompson group is context-free. Bull. Lond. Math. Soc. **39**(2), 235–241 (2007). https://doi.org/10.1112/blms/bdl043
19. Lindenmayer, A.: Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. J. Theoret. Biol. **18**(3), 280–99 (1968). https://doi.org/10.1016/0022-5193(68)90079-9

20. Muller, D.E., Schupp, P.E.: The theory of ends, pushdown automata, and second-order logic. Theoret. Comput. Sci. **37**(1), 51–75 (1985). https://doi.org/10.1016/0304-3975(85)90087-8
21. Nekrashevych, V.: Self-similar Groups, Mathematical Surveys and Monographs, vol. 117. American Mathematical Society, Providence (2005). https://doi.org/10.1090/surv/117
22. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer, Berlin (1997). https://doi.org/10.1007/978-3-642-59126-6
23. Rozenberg, G.: Extension of tabled OL-systems and languages. Int. J. Comput. Inf. Sci. **2**, 311–336 (1973)
24. Sidki, S.: Automorphisms of one-rooted trees: growth, circuit structure, and acyclicity. J. Math. Sci. (New York) **100**(1), 1925–1943 (2000). https://doi.org/10.1007/BF02677504. Algebra, 12