Lamplighter Groups and Automata

Sanjay Jain
Department of Computer Science
National University of Singapore
Singapore 117417
Email: sanjay@comp.nus.edu.sg

Birzhan Moldagaliyev
Faculty of Engineering and Natural Sciences
Suleyman Demirel University
1/1 Abylai Khan Street, Kaskelen, Almaty
Kazakhstan, 040900
Email: birzhanm@gmail.com

Frank Stephan
Departments of Mathematics and of Computer Science
National University of Singapore
Singapore 117690
Email: fstephan@comp.nus.edu.sg

Tien Dat Tran
Department of Computer Science
National University of Singapore
Singapore 117417
Email: tran.tien.dat@u.nus.edu

***Abstract*—This paper investigates presentations of lamplighter groups using computational models from automata theory. The present work shows that if G can be presented such that the full group operation is recognised by a transducer, then the same is true for the lampglighter group $G \wr \mathbb{Z}$ of G. Furthermore, Cayley presentations, where only multiplications with constants are recognised by transducers, are used to study generalised lampglighter groups of the form $G \wr \mathbb{Z}^d$ and $G \wr F_d$, where F_d is the free group over d generators. Additionally, $\mathbb{Z}_k \wr \mathbb{Z}^2$ and $\mathbb{Z}_k \wr F_d$ are shown to be Cayley tree automatic.**

***Acknowledgments*—Sanjay Jain and Frank Stephan have been supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112. In addition Sanjay Jain was also supported in part by NUS grant C252-000-087-001. The paper was developed during the final year project of Tien Dat Tran [25] and Birzhan Moldagaliyev's PhD thesis [20]. The authors would like to thank Dmitry Berdinsky and Bakhadyr Khoussainov for correspondence.**

I. Introduction

Mathematics has a long history of ways to represent groups. From the viewpoint of computer science and logic, in particular the ways of representing groups using finite automata are interesting, as finite automata allow many operations and decision procedures due to their simple form. The first approach was by Hodgson [14], [15] and this approach was rediscovered and extended by Khoussainov and Nerode [19] as well as Blumensath and Grädel [8], [9]. The common idea of these researchers is to generalise the notion of a finite automaton on words to a finite automaton on relations where all inputs are read at the same speed, one symbol per cycle, with a special symbol marking exhausted inputs (so that from them still a symbol is read per cycle). This approach is quite fruitful and the structures defined this way have, as long as the logical language is finite, a decidable first-order theory [15], [19]. Relations recognised by automata are called automatic and in this model it even holds that relations first-order defined with automatic parameters are again automatic, where the automaton recognising the newly defined relation can be computed from the automata used in the parameters of the formula as well as the formula itself. Automatic functions are functions where the graph is an automatic relation. If automatic functions have one input and one output, then they coincide with the functions computed by a one-tape Turing machine in linear time where input and output start at the same position, this additional constraint of the output replacing the input from the same symbol onwards is needed for the result [10]. The second approach towards automatic groups is due to Epstein, Cannon, Holt, Levy, Paterson and Thurston [12] who wrote a whole textbook on the topic; while Hodgson required that the full group operation is automatic, Epstein, Cannon, Hold, Levy, Paterson and Thurston only require that if one fixes the second input as a constant, then the derived unary functions are automatic. Due to the fact that concatenations of automatic functions are automatic, it is sufficient to require this for constants being a generator in the case of a finitely generated group and the authors consider indeed only those groups. While this generalises the concepts, they chose further restrictions in order to obtain natural representatives of the group members, the so called normal forms. Each group member is represented by a word over a finite alphabet representing some finite set of generators of the group and furthermore, not all representatives are used, but only those in a given regular set such that the equality restricted to this set of representatives is automatic. Khoussainov, Kharlampovich and Miasnikov [18] generalised this work by keeping the requirements of the group being finitely generated and the group operation being automatic only for multiplication with constants; however, they allowed any regular set of representatives, no longer requiring that they are words over an alphabet consisting of generators of the group. Any other finite alphabet can be used instead.

In our work, we want to explore an alternative generalisation, namely we are more faithful to the approach of Hodgson [14], [15] and require that the full group operation is recognised by a finite automaton. However, we do this not in the framework of automatic functions but instead in the framework of transducers which can be nondeterministic and which can read all inputs at different speeds. We in particular show that recognisability by transducers in this framework inherits from a group G to its lamplighter expansion $G \wr \mathbb{Z}$ and this result is then for us a motivation, to study the representability of lamplighter groups in general.

There are well-known examples for monoids and semigroups where the full group operation is recognised by transducers and not by finite automata; for example, the monoid of binary strings with concatenation is recognisable by a transducer without any equivalent automatic model [23, Corollary 3.45]. However, there are no similarly prominent examples on the level of groups. Most research with respect to transducers and groups is done in the framework of Epstein, Cannon, Holt, Levy, Paterson and Thurston [12] and then only multiplication with constants is recognised by transducers. Therefore we believe that our results on groups with fully transduced group operation provide new insights into the field. In particular the lamplighter construction $G \mapsto G \wr \mathbb{Z}$ of a transducer presented group provides a very natural construction to obtain such groups; further examples stem from the class of Baumslag-Solitar groups. We also show that for every finite group G, its two-dimensional lamplighter group $G \wr \mathbb{Z}^2$ is Cayley tree automatic. Furthermore, we investigate other automata-theoretic properties of groups of the form $G \wr \mathbb{Z}^d$ for general d and show that they are Cayley transducer recognisable. Berdinsky and Khoussainov [5], [6] studied recognisability of groups by pushdown automata. We show that many of their examples are Cayley tree automatic and can then with a general theorem, Theorem 24, also derive that these structures can also be presented by pushdown automata.

Our results aim towards a better understanding of the connections between group theory and finite automata / transducers. These are fundamental concepts in computer science and representability of structures by automata has been a widely studied topic with connections to mathematical logic and the decidability of the corresponding structures.

II. Transducer Presentations of Groups

In the following, we want to explain our concepts a bit more in depth. A finite automata presentation $(D, \circ)$ of G can be thought as a surjective group homomorphism $\phi : D \to G$ where D is a regular language and $\circ_D$ is an automatic function. Usually ϕ is assumed to be injective. However, some authors loosen the requirement of injectivity, instead require equivalence of two elements x and y such that $\phi(x) = \phi(y)$ to be automatic. In the case of finite automata recognisability, these two conditions are equivalent. However, for more expressive computational machines, injectivity is often strictly stronger requirement. In this formalization all necessary information about G comes together in finite automata recognising D and $\circ$.

Apart from finite automata presentations there are Cayley automatic presentations. A finitely generated group G with a set S of generators is said to be Cayley automatic if there is a group isomorphism $\phi : D \to G$ such that for a generator $s \in S$, the corresponding multiplication $x \to x \circ_D \phi^{-1}(s)$ is automatic function with domain and range D.

The notion of an asynchronous transducer predates that of a synchronous finite automaton; however, it does not share the decidability properties of the latter. While synchronous finite automata have to read input from all tapes synchronously, a transducer may read input tapes in any order. For this reason, transducer recognisable relations generalise automatic relations. Recall that to define an automatic relation, one has to ensure that all inputs are of the same length by filling shorter input tapes with some special symbol, say $\#$. For transducers there is no need to ensure that inputs are of same length. The downside working with transducer recognisable relations is somewhat limited closure properties compared to regular languages. Let us mention definitions and some properties of transducers and relations they induce. Below Σ denotes some finite alphabet, and ε denotes an empty string.

Definition 1 (Transducer). *A transducer $M = (Q, \Sigma, \delta, s_0, F)$ is a nondeterministic finite automaton having multiple tapes, say n, so that it reads them asynchronously. Correspondingly, a transition function for standard finite automata $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, is replaced by $\delta : Q \times \Sigma_\varepsilon^n \to \mathcal{P}(Q)$, where $\Sigma_\varepsilon = \Sigma \cup \varepsilon$.*

Observe that by adding few extra states to M, one could construct an equivalent transducer M' so that it reads exactly one symbol per transition. A notion of transducer gives rise to transducer recognisable relations.

Definition 2 (Transducer recognisable relations and functions). *A relation $R \subseteq \Sigma^n$ is said to be transducer recognisable if there is a transducer recognising R. A function f is said to be transducer recognisable if its graph, $graph(f) = \{(x, y) : f(x) = y\}$, forms a transducer recognisable relation.*

Transducer recognisable relations are also known as weakly-regular predicates. Monks [21] investigates the closure properties of weakly regular predicates, showing the following results.

Proposition 3 (Monks [21]). *Weakly regular n-ary predicates P, Q satisfy the following statements:*

- *The predicate $\neg P$ is not necessarily weakly regular;*
- *The predicate $P \wedge Q$ is not necessarily weakly regular;*
- *The predicate $\forall x_1 P(x_1, \ldots, x_n)$ is not necessarily weakly regular;*
- *The predicate $\exists x_1 P(x_1, \ldots, x_n)$ is weakly regular;*
- *The predicate $P \vee Q$ is weakly regular.*

These closure properties are in stark contrast with automatic relations, where automatic relations are closed with respect to all of above operations. On the positive side, transducer recognisable relations are closed under union and taking existential quantifier. Furthermore, transducer recognisable functions have the additional closure properties listed below.

Proposition 4 (Berstel [7], Chapter III.4). *Let f be a transducer recognisable bijective function. Then its inverse f^{-1} is also transducer recognisable.*

Proposition 5 (Elgot and Mezei [11]; Berstel [7]). *Let D be a regular language together with transducer recognisable functions $\{f_i : D \to D\}_{i=1}^k$ and $g : D^k \to D$. Then k-ary composition of functions $\{f_i\}$ over g, $g(f_1, \ldots, f_k) : D^k \to D$ is also transducer recognisable.*

It is easy to see that the above theorem can be generalized to the case when $f_i : D^{m_i} \to D^{n_i}$ for each i. In that case $g : D^N \to D^r$, where $N = \sum_{i=1}^k n_i$ and $g(f_1, \ldots, f_k) : D^M \to D^r$, where $M = \sum_{i=1}^k m_i$.

A. Automatic Groups and Cayley Automatic Groups

Let ρ be some automata-theoretic notion of computation such as transducers or tree automata. We say that a group $(G, \circ)$ forms a ρ-recognisable structure if $(G, \circ)$ is isomorphic to $(H, \circ)$ with

1. H is a ρ-recognisable set;
2. The equality of objects in H is a ρ-recognisable relation;
3. The group multiplication function $\circ : H \times H \to H$ is a ρ-recognisable function.

The above conditions say that group H is recognisable by ρ in somewhat global sense, so that multiplication of any two elements is ρ-recognisable. If one weakens this condition, one receives the notion of Cayley automatic groups:

3. For every fixed $h \in H$, the mapping $x \mapsto x \circ h$ from H to H is a ρ-recognisable function.

Furthermore, note that for Cayley automatic groups, Khoussainov, Kharlampovich and Miasnikov [18], who introduced this notion, did only consider finitely generated groups so that they could restrict themselves to postulating the above condition only for members of the set of generators of the group.

Equivalence of left and right multiplications. In above definition of Cayley ρ-recognisable groups, we have used the right multiplication by a group generator. One could also use left multiplication. It turns out that resulting definitions are equivalent up to group isomorphism. In other words, for any group that is Cayley ρ-recognisable with respect to left multiplication there is isomorphic copy which is Cayley ρ-recognisable with respect to right multiplication and vice versa. Let us provide a short argument for this fact.

Short argument. Suppose that $(G, \circ)$ is left Cayley transducer presented group with a generating set S. The given condition says that a function $x_s : g \to s \circ g$ is ρ-recognisable for any $s \in S$. Consider the opposite group of G, $(G^{op}, \star)$, which has same underlying set with G, but reversed group operation. In other words, for any $x, y \in G^{op}$, $x \star y = y \circ x$. Observe that S also generates G^{op}. Furthermore, for any $s \in S$, a right multiplication map by s in G^{op} is same as left multiplication by s in G and thus, the mapping

$$x_s : g \to g \star s = s \circ g$$

is transducer recognisable by definition. Finally, observe that $G \simeq G^{op}$ via inverse map $x \to x^{-1}$. Thus $G^{op} \simeq G$ is right Cayley transducer presented group. In the similar fashion, one could show that for any right Cayley ρ-recognisable group there is isomorphic left Cayley ρ-recognisable group.

B. Formal Definitions

Given any automata-theoretic notion of recognisability ρ, transducers and tree automata in our case, we are going to define a general notion of ρ group presentation and Cayley ρ group presentation.

Definition 6. *A ρ presentation of a group G is a surjective group homomorphism $\phi : D \to G$, such that the following conditions hold:*

- *D is ρ recognisable group;*
- *$L_= = \{(x, y) \in D \times D : \phi(x) = \phi(y)\}$ is ρ recognisable;*
- *Some ρ recognisable function computes, for inputs $x, y \in D$, a $z \in D$ with $\phi(x) \circ \phi(y) = \phi(z)$.*

If ϕ happens to be bijective, then the presentation is said to be one-to-one. A group G is said to be ρ presented if it has ρ-presentation.

Next we describe definition of Cayley presented finitely generated groups.

Definition 7. *A Cayley ρ presentation of a group G is a surjective group homomorphism $\phi : D \to G$, where*

- *D is Cayley ρ recognisable group;*
- *$L_= = \{(x, y) \in D \times D : \phi(x) = \phi(y)\}$ is ρ recognisable;*
- *For all fixed y of D, there is a ρ recognisable function which maps every $x \in D$ to a $z \in D$ with $\phi(z) = \phi(x) \circ \phi(y)$.*

If ϕ happens to be bijection, then the presentation is said to be one-to-one. A group G is said to be Cayley ρ presented if it has Cayley ρ presentation.

C. Basic Properties

Let us list down some of the immediate properties of newly defined transducer presented and Cayley transducer presented groups.

Inclusions. As we have mentioned earlier transducer presentations and tree automata presentations imply Cayley transducer presentations and Cayley tree automata presentations respectively. Since both transducers and tree automata generalize finite automata. We have that automatic groups are both transducer presented and tree automata presented.

Transducer presented groups. In the context of groups there could be two underlying structures $(G, \circ)$ and $(G, \circ, \cdot^{-1})$, where the latter adopts inverting as an additional unary function. For automatic structures, finite automata recognizability for both structures are equivalent. One could ask if the same is true for transducer presented groups. In other words, given a transducer presented group G, is taking inverse a transducer recognisable operation? It turns out to be so for those groups with one-to-one presentations.

Proposition 8. *For a transducer one-to-one presented group G, an inverting map $x \to x^{-1}$ is transducer recognisable in underlying presentation.*

Proof. Let $(M, \circ_M)$ be underlying transducer recognisable structure so that $M \simeq G$. We need to show that the inverse map $m \to m^{-1}$ is transducer recognisable, that is, $R = \{(m, m^{-1}) : m \in M\}$ is transducer recognisable relation. Let T be a transducer recognising group multiplication in M. Then T has three components corresponding $x \circ_M y = z$. To build a transducer P recognising R we simulate T with input m in the first component and the unique identity element id_M in the third component. Then a transducer nondeterministically guesses the second component so that T accepts the given configuration, thus identifying m^{-1}. Hence, R is transducer recognisable. □

One could show that for given n, multiplication of n elements of G is transducer recognisable.

Proposition 9. *For transducer presented group G, n-ary group multiplication is transducer recognisable for any n.*

Proof. Let M be an underlying transducer presentation of G. We need to show that an n-ary multiplication

$$f_n(x_1, \dots, x_n) = x_1 \circ_M x_2 \circ_M \dots \circ_M x_n$$

is transducer recognisable. The case $n = 1$ corresponds to the case of identity map, which is clearly transducer recognisable. The case $n = 2$ corresponds to the definition of transducer recognizability for M. We use induction to show the claim for general n. Suppose that it holds for k, that is, f_k is transducer recognisable. Observe that f_{k+1} can be constructed as k-composition of f_2 and $k-1$ pieces of f_1 as follows:

$$f_{k+1}(x_1, \dots, x_{k+1}) = f_k(f_2(x_1, x_2), f_1(x_3), \dots, f_1(x_{k+1})).$$

By closure of transducer recognisable functions under n-ary compositions mentioned in the background section, we conclude that f_{k+1} is transducer recognisable. □

Let us look at the notion of regular pattern.

Definition 10 (Regular patterns)**.** *Let $(G, \circ)$ be a group. A regular pattern is a function $p : G^n \to G$, $p(x_1, x_2, \dots, x_n) = a_0 \circ x_1^{b_1} \circ a_1 \circ x_2^{b_2} \circ \dots \circ a_{n-1} \circ x_n^{b_n} \circ a_n$, where a_i's are constants in G and $b_i \in \{-1, +1\}$.*

Proposition 11. *For a transducer one-to-one presented group G a regular pattern is transducer recognisable in the underlying presentation.*

Proof. Let M be an underlying transducer presentation of G so that $M \simeq G$. Let $\{a_i\}_{i=0}^n$ be fixed elements of M and $\{b_i\}_{i=1}^n$ be power signs from $\{\pm 1\}$. Observe that for each i with $0 \le i \le n$, a constant map $f_i : M \to M$ given as $f_i(m) = a_i$ is transducer recognisable. Moreover for each j with $1 \le j \le n$, a map $h_j : D \to D$ given as $h_j(m) = m^{b_j}$ is transducer recognisable, simply because it is either identity map or inverse map, both of which are transducer recognisable maps.

Consider now a multiplication map for $2n+1$ elements, $g : D^{2n+1} \to D$ such that $g(x_1, \dots, x_{2n+1}) = x_1 \circ_M x_2 \circ_M \dots \circ_M x_{2n+1}$. By the previous proposition g is transducer recognisable function. By composing functions $\{f_i\}$ and $\{h_j\}$ over g one obtains a transducer recognisable function $\bar{g} = g(f_0, h_1, f_1, \dots, h_n, f_n) : D^{2n+1} \to D$. Now observe that $p(x_1, \dots, x_n) = \bar{g}(id_M, x_1, id_M, x_2, \dots, id_M, x_n, id_M)$. Hence, a computation of p reduces to computation of $\bar{g}$. This shows that p is indeed transducer recognisable. □

D. Algorithmic Problems

As in the case of automatic groups, the word problem for both Cayley transducer presented groups and Cayley tree automata presented groups is solvable. Let ρ denote either transducer or tree automata recognisability in the following proof.

Proof. Let G be a Cayley ρ presented group with corresponding presentation $\phi : D \to G$. Suppose that S is a generating set for G and let $w = s_1 s_2 \dots s_n$ be a word of length n in S. The objective is to check if the group element corresponding to w equals to the identity element, that is, $w|_G = id_G$. This problem can be solved by pulling elements of G back to D. For each s_i, there is a corresponding d_i such that $\phi(d_i) = s_i$. Then verification of $w|_G = id_G$ reduces to the verification that $d_1 \circ d_2 \circ \dots \circ d_n \in \ker \phi$ or $(d_1 \circ \dots \circ d_n, id_D) \in L_=$. Since computations of both $d_1 \circ \dots \circ d_n$ and membership in $L_=$ are ρ recognisable. It follows that the mentioned word problem is solvable. □

III. Lamplighter Groups

The original lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}$ models a lamplighter on an infinitely long road who can go forward or backward on the road and switch lamps on and off; as initially, all lamps are off, there are only finitely many lamps on at any given time. Grigorchuk and Żuk [13] realised this group as an automaton group with a two-state automaton; Khoussainov, Kharlampovich and Miasnikov [18] showed the the lamplighter group is Cayley automatic and Berdinsky [5] showed that it is Cayley biautomatic. We study lamplighter groups in other automatic frameworks and in particular look at the more general definition of $G \wr H$ where G is ranging over groups easily representable by automata (or even only over finite groups) and H is either $\mathbb{Z}^d$, the d-dimensional commutative version of $\mathbb{Z}$, or F_d, the d-dimensional noncommutative version of $\mathbb{Z}$, that is, the free group with d generators. Here $G \wr H$ is the restricted wreath product. Berdinsky [5] had studied these groups before with respect to their representability by pushdown-automatic structures.

Definition 12 (Lamplighter group)**.** *Given groups G, H, a lamplighter group of G over H is given by a restricted wreath product $G \wr H$.*

Let us expand a little on above definition. Set a group $K = \bigoplus_H G$ to be a direct sum of G over H. Each element $k \in K$ can be viewed as a function $k : H \to G$ with a finite support. There is a natural action of H on $Aut(K)$ given by

$$hk(x) = k(h^{-1} \circ x)$$

where $k \in K$ and $x, h \in H$. Denoting above action θ, a restricted wreath product of G over H is then a semidirect product of K and H over group action θ, $G \wr H = K \rtimes_\theta H$. Since canonical action θ is mostly assumed, one could drop θ, and simply write $G \wr H = K \rtimes H$. Observe that each element of $G \wr H$ can be considered as a pair (k, h) with $k \in K$ and $h \in H$. The group multiplication rule for two such pairs (k_1, h_1) and (k_2, h_2) is given as

$$(k_1, h_1) \circ (k_2, h_2) = (k_1 \circ_K h_1 k_2, h_1 \circ_H h_2).$$

A. Representing Lamplighters with Transducers

Let S_G and S_H be sets of generators for G and H and let $k_g \in K$ such that

$$k_g(x) = \begin{cases} g, & \text{if } x = id_H; \\ id_G, & \text{otherwise.} \end{cases}$$

Then a collection $\{(k_g, id_H) : g \in S_G\} \cup \{(id_K, h) : h \in S_H\}$ generates $G \wr H$. This means that $G \wr H$ is finitely generated whenever G and H are finitely generated. As for finite presentability, there is a famous result of Baumslag [3], which asserts that $G \wr H$ is finitely presented if and only if either G is trivial group or H is finite.

Lamplighter groups are an interesting class of groups that have been studied in geometric group and automata theory [1], [5], [6], [13], [18]. First, we give a description of the simplest lamplighter group $L(\mathbb{Z}_2)$, where $L(G)$ refers to $G \wr \mathbb{Z}$ throughout the paper. We can view each element in $L(\mathbb{Z}_2)$ to be the action of a lamplighter on an infinite street with lamps at integral coordinates. The lamplighter starts at where he was resting previously and walking along the street, turning the switch of finitely many lamps and then stops to rest at one lamp. Initially, all the lamps are off and the lamplighter rests at the origin. Each element can be represented by a finite sequence of 0’s and 1’s, where 1 represents turning the switch, with a origin marker ’, and a lamplighter marker ” for the stopping position. For example, 100’10”1 represents that the lamp lighter turns the switch at position -3, 0, 2 and then stops at a position that is 2 steps forward / to the right of his original position. Let $a, b \in L(\mathbb{Z}_2)$, we define the group operation $a \circ b$ to be the net effect of the lamplighter after carrying out a and then b. We note that the lamplighter group $L(\mathbb{Z}_2)$ is finitely generated, and the generators are: ’”1 (turn the switch at current position and does not move), ’0” (move one step to the right) and ”0’ (move one step to the left).

The presentation we discussed earlier for $L(\mathbb{Z}_2)$ gives us a transducer recognisable group. We just need to require that there are not leading or trailing 0’s to ensure that each element has a unique presentation. The group operation can be done by a transducer by first aligning the ” (stopping) marker of the 1st operand with the ’ (origin) marker of the 2nd operand, and then carrying out component-wise addition in $\mathbb{Z}_2$. The transducer then copies the ’ marker of the 1st operand and the ” marker of the 2nd operand to the result. Leading and trailing 0’s are also deleted. An example is given below:

1	0	0	’0	1	1	”0	0	1
			1	0	0	’1	0	”1
1	0	0	’1	1	1	1	0	”

We also observe that the base group of the lamplighter group does not need to be $\mathbb{Z}_2$, but could be any group G. In the following result we show that transducer presentability is closed under taking lamplighter group over $\mathbb{Z}$.

Theorem 13. *Given a transducer one-to-one presented group G, its lamplighter group $L(G)$ is also transducer one-to-one presented.*

Proof. The idea is to come up with transducer presentation for $L(G)$ starting from the transducer presentation of G. Without a loss of generality, we can assume that G itself is transducer recognisable. In order to construct a transducer presentation of $L(G)$, we add following additional symbols to the alphabet used to present G:

- @: to separate the elements of G in our presentation of $L(G)$;
- @’: to mark the origin / original position of the lamplighter;
- @”: to mark the stopping position of the lamplighter;
- @’”: used when the origin coincides with the stopping position.

The transducer can read inputs at different speeds, so it can align @” of the 1st operand with @’ of the 2nd operand, as well as aligning the other @ such that the components of the 1st operand is aligned with the corresponding components in the 2nd operand. With such alignment, component-wise application of the group operation $\cdot$ can also be done since $(G, \cdot)$ is transducer-recognisable. The last step would be to copy @’ of the 1st operand and @” of the 2nd operand to the result, and we have constructed a transducer to recognise the group operation for $L(G)$. $\square$

To illustrate the construction in Theorem 13, we can look at the following example of multiplying elements in $L(\mathbb{Z})$. Note that the number are written from most to least significant digit for easier human reading, while in actual fact they should be written from the least significant digit.

@	-	2	@”	0	@’		8	@	3	@
			@’	3	@”		3	@	3	@
@	-	2	@	3	@’”	1	1	@	6	@

If we fix the position of @’ and @” to always coincide, then the elements in this subgroup of $L(G)$ is isomorphic to the group of functions from $\mathbb{Z}$ to G which is id_G at almost everywhere with component-wise addition (here id_G denotes the neutral element of G and addition is the group operation $\cdot$ of G). The set of presentation for this subgroup is also regular, so this subgroup is transducer-recognisable. Hence we have the following corollary.

Corollary 14. *Let G be a transducer one-to-one presented group. Then the direct sum of G over $\mathbb{Z}$, $\bigoplus_{\mathbb{Z}} G$, is also transducer presented.*

B. Transducer Presentations and Baumslag-Solitar Groups

Apart from Lamplighter groups, a special case of Baumslag-Solitar groups, $BS(1, n) = \langle a, b : ab = ba^n \rangle$ is also transducer presented.

Theorem 15. *The group $G = \langle a, b : ab = ba^n \rangle$ is transducer one-to-one presented.*

Proof. First, we show that this group has a simple normal form for all its elements. All elements of G can be uniquely expressed as $b^i a^k b^{-j}$ where $i, j \geq 0$ and $ij = 0$ if $n \mid k$.

The second condition is necessary to ensure uniqueness. $ab = ba^n \implies a = ba^n b^{-1}$. Multiply by m times we have:

$$a^m = (ba^n b^{-1})(ba^n b^{-1}) \dots (ba^n b^{-1}) = ba^{mn} b^{-1}.$$

In other words, if index k of a is divisible by n, we can always reduce k to k/n and decrement i and j. To make the normal form unique, we need $i = 0$ or $j = 0$ in this case, so that we cannot decrement them.

Repeatedly applying the previous equation, we get the useful formula

$$\begin{aligned} a^m &= ba^{mn} b^{-1} = b(ba^{mn^2} b^{-1}) b^{-1} \\ &= b^2 a^{mn^2} b^{-2} = \dots = b^h a^{mn^h} b^{-h}. \end{aligned}$$

This fact will be used later to help us compute the normal form of the product of 2 elements. It is obvious that all the generators of the groups, a, b, a^{-1}, b^{-1} can be expressed in the normal form. Hence, if we can show that the product of 2 elements of normal form can also be expressed in normal form, then all elements of G can be expressed in the mentioned normal form.

Consider $(b^i a^k b^{-j})(b^{i'} a^{k'} b^{-j'}) = b^i a^k b^{i'-j} a^{k'} b^{-j'}$. If $i' - j \geq 0$, we replace a^k by $b^{i'-j} a^{kn^{i'-j}} b^{j-i'}$ and get:

$$\begin{aligned} (b^i a^k b^{-j})(b^{i'} a^{k'} b^{-j'}) &= b^i a^k b^{i'-j} a^{k'} b^{-j'} \\ &= b^i b^{i'-j} a^{kn^{i'-j}} b^{j-i'} b^{i'-j} a^{k'} b^{-j'} \\ &= b^{i+(i'-j)} a^{kn^{i'-j}+k'} b^{-j'} \end{aligned}$$

which is of normal form. Similarly, if $i' - j < 0$, we apply the formula to $a^{k'}$ and get

$$(b^i a^k b^{-j})(b^{i'} a^{k'} b^{-j'}) = b^i a^{k' n^{j-i'} + k} b^{-(j'+(j-i'))}.$$

We can turn this normal form into a transducer-recognisable presentation for G. Each element of G can be identified by 3 numbers i, j and k. We encode k in base n, written from the least significant digit, put a marker $'$ after i digits and a marker $''$ after j digits. For example, let's say $n = 2$, the element $b^3 a^{13} b^{-5}$ will be presented as

$$\underbrace{\overbrace{1\ \ 0\ \ 1'}^{i}\ \ 1\ \ \#''}_{j}$$

where 1101 is the base 2 representation of 13, and $\#$ is the symbol used after the digits of k have been exhausted.

Given the presentation of $b^i a^k b^{-j}$ and $b^{i'} a^{k'} b^{-j'}$, we need to compute the presentation of the product $b^{i''} a^{k''} b^{-j''}$ using a transducer. This can be done by aligning the j marker with the i' marker, carrying out normal base n addition digit by digit and copying over the i marker to i'' and j' marker to j'' in the process.

Let's look at a more concrete example to understand how this work. Now, let $n = 2$, we need to compute the product of $b^3 a^{13} b^{-5}$ and $b^7 a^5 b^{-4}$, which is $b^5 a^{57} b^{-4}$. Here, $i' - j \geq 0$ so we need to compute $i'' = i + (i' - j)$, $j'' = j'$ and $k'' = kn^{i'-j} + k'$. Now consider the pictorial view

$$\begin{array}{ll} \underbrace{\qquad}_{i'-j}\underbrace{\overbrace{1\ \ 0\ \ 1'}^{i}\ \ 1\ \ \#''}_{j} & b^3 a^{13} b^{-5} \\ \overbrace{\underbrace{1\ \ 0\ \ 1\ \ \#''}_{j'}\ \ \#\ \ \#\ \ \#'}^{i'} & b^7 a^5 b^{-4} \\ \overbrace{\underbrace{1\ \ 0\ \ 0\ \ 1''}_{j''}\ \ 1'}^{i''}\ \ 1\ \ \# & b^5 a^{57} b^{-4} \end{array}$$

and, as seen in the diagramme, the first input is shifted to the right by $(i' - j)$, effectively multiplying k by $n^{i'-j}$ and adding $(i' - j)$ to i. Hence, in the result, $k'' = kn^{i'-j} + k'$ and $i'' = i + (i' - j)$. The case for when $i' - j < 0$ is also similar.

It should be noted that a key step in this computation is in aligning the j marker with the i' marker. This can be done by the transducer by nondeterministically guessing $(i' - j)$ and then reading that many symbols from the 2nd input without reading anything from the 1st input. A finite automaton must read all the inputs at the same speed, thus it cannot perform such alignment of input.

However, if we fix one operand to be a constant, then the resulting function is automatic. For example, if we fix the right operand to be $c \in G$, we can construct an automata to compute $f_c : G \to G, x \mapsto xc$, that is, given the presentation of $b^i a^k b^{-j}$, we need to compute the presentation of the product $(b^i a^k b^{-j})(b^{i'} a^{k'} b^{-j'}) = b^{i''} a^{k''} b^{-j''}$ where i', j' and k' are constants. Note that in the original transducer to compute the group operation in general, the only step that an automaton cannot do is aligning the j marker with the i' marker, that is, aligning the two inputs of variable length, which requires the two inputs to be read at different speed. However, the product input is always read together with either one or both of the operand input, that is, writing the result can be done by an automaton. Now when we fix one operand, the problem of aligning disappears because one operand has constant length, we can use nondeterminism to guess the length of the variable operand to align them using an automaton. Thus, fixing one operand of this particular transducer presented group will result in an automatic function. $\square$

Remark 16. *It is worth mentioning that both group classes discussed above are not automatic. A simple way to verify this can be done by growth rates in groups. Oliver and Thomas [22] have shown that automatic groups have polynomial growth. As for Baumslag Solitar groups $B(m, n)$ with $m < n$ and Lamplighter groups $L(G)$ with nontrivial G, it is well known that they are of exponential growth.*

C. Cayley Transducer Presented Lamplighter Groups

The 2-dimensional lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^2$ is similar to the 1-dimensional, but instead of an infinite street, the lamps are located at integral coordinates in a 2D-plane. Each element in the group representing the action of the lamplighter turning the switches of finitely many lamps and then stops to rest at one of the lamp. We can represent the elements of the group as a regular language using the symbols $0, 1, \bullet, *, x, |$ as followed:

- 0 and 1 represents whether the lamp at the coordinate is off or on respectively; $|$ is the line separator; $\bullet$ is the origin marker, placed to the left of the symbol for the lamp at (0, 0); $*$ is the destination marker, indicating where the lamplighter stops. In addition, on rows other than where the lamplighter stops, x indicates the destination marker projected to that row, that is, if the destination marker $*$ is at (a, b), then we put a projected destination marker x at (i, b) for all $i \neq a$.
- Each row of an element in the 2D lamplighter group is represented in a similar way to 1D lamplighter group (for example, deleting leading and trailing zeroes unless they are required to mark the origin/destinations). However, only 1 row will have the origin marker $\bullet$.
- The first row to be represented is the one with smallest row number such that either a lamp is switch on that row, or the origin is on that row, or the destination is on that row. The last row is the one with the highest row number with either of the 3 afore-mentioned conditions.

For example, the string $1 \bullet 0x0|x0| * 0001|1x001$ represents the following element:

	-1	0	1	2	3	4
0	1	$\bullet 0$	$x0$	0	0	0
1	0	0	$x0$	0	0	0
2	0	0	$*0$	0	0	1
3	0	1	$x0$	0	1	0

The lamps at (0, -1), (2, 4), (3, 0) and (3, 3) are switched. The lamplighter stops at (2, 1).

D. Cayley Presentability

Theorem 17. *The 2D lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^2$ is Cayley transducer one-to-one presented.*

Proof. The representation described earlier allows multiplication with a constant on the right to be transducer-recognisable. Note that left multiplication with a constant can also be made transducer-recognisable in a similar way by projecting the origin markers onto the rows instead of the destination marker.

To show that right multiplication with constant is transducer recognisable, we first observe that the 2D lamplighter group is finitely generated from $\bullet * 1$ (switch current lamp), $*0 \bullet 0$ (move left), $\bullet 0 * 0$ (move right), $*0|\bullet x0$ (move up) and $\bullet x0|*0$ (move down). Hence, to show that right multiplication with constant is transducer-recognisable, we just need to show that right multiplication with each of the generators is transducer-recognisable.

Switching current lamp is easy, the transducer just need to change $*0$ to $*1$ and vice versa.

For moving left, the transducer uses asynchrony to align the line separators $|$ and process each row separately. It uses non-determinism to guess the row that has the destination marker $*$. For that row, it uses non-determinism to guess the position of $*$ and moves it to the left, possibly adding an extra 0 on the left if $*$ is already on the leftmost position, that is, change $*100$ to $*0100$ and so on. For other rows, the transducer does the same thing, but with the projected destination marker x instead of $*$. Moving right is done in a similar way.

For moving up, the transducer uses non-determinism to guess the row that has the destination marker. If it is the 1st row, the transducer prepends $*0|$. Otherwise, at the 1 row above the row of the old destination marker, it changes x to $*$. Finally, at the row of the old destination marker, $*$ is changed to x. Moving down is done in a similar way. □

In fact, this method can be used to show that the n-dimensional lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^n$ has transducer-recognisable constant multiplication.

Theorem 18. *The n-dimensional lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^n$ is Cayley transducer one-to-one presented.*

Proof. We will use induction. The base case is established by the previous proposition. Supposed that the $(n-1)$-dimensional lamplighter group has transducer-recognisable constant multiplication by using separator $|_{n-1}$ and destination marker x_{n-1}. To represent the n-dimension, we will use the symbol $|_n$ to separate the $(n-1)$-dimension hyperplane, and project the destination marker x_n into each $(n-1)$-dimension hyperplane as x_{n-1}, that is, if x_n is at $(a_1, \ldots, a_n)$ then x_{n-1} is at $(a_1, \ldots, a_{n-1}, i)$ for all $i \neq a_n$. In each $(n-1)$-dimension hyperplane, we use the representation that is transducer-recognisable (by inductive hypothesis).

Similar to the 2D case, n-dimensional lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^n$ is finitely generated from switching current lamp and moving unit distance in each dimension. Switching current lamp is easily transducer-recognisable. Moving in the 1st $(n-1)$ dimensions can be done by moving in each $(n-1)$-dimension hyperplane, which by the inductive hypothesis is transducer-recognisable. Moving in the nth dimension is also transducer-recognisable, by using non-determinism to guess the hyperplane containing x_n, which is changed to x_{n-1} and in the next/previous hyperplane changing x_{n-1} to x_n (or add an empty hyperplane if necessary).

Hence, we have showed that the n-dimensional lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^n$ has transducer recognisable constant multiplication, hence Cayley transducer presented. Furthermore, the given presentation is one-to-one. □

E. Cayley Presentations of $G \wr \mathbb{Z}^n$

Even though we used the simple $\mathbb{Z}_2$ group as the base for the lamplighter group in the previous constructions, it can be replaced with any group that is Cayley one-to-one presented. The construction is very similar, because a transducer can use asynchrony to align the elements correctly. Thus, the slightly more general theorem follows.

Theorem 19. *Let G be Cayley transducer one-to-one presented group. Then $G \wr \mathbb{Z}^n$ is also Cayley transducer one-to-one presented.*

Berdinsky [5, Theorem 4.3.1] showed that when G is Cayley automatic, so is $G \wr \mathbb{Z}$. It is an open question whether the same is true for semiautomatic groups as introduced by Jain, Khoussainov, Stephan, Teng and Zou [16], [17]. The same is, to our knowledge, open for Cayley biautomatic groups. For finding counter examples for a negative solution to these questions, note that finitely generated Cayley automatic groups are semiautomatic [16], thus a negative solution, if found, must stem from semiautomatic groups G which are not Cayley automatic.

IV. Tree Automatic Groups and Lamplighters

Let us first outline a definition of tree automata.

Definition 20 (Tree Automata)**.** *A (non-deterministic) leaf-to-root tree automaton is a tuple $(Q, \mathcal{F}, Q_f, \delta)$. Q is a set of states. $\mathcal{F}$ is a set of ranked alphabet, that is, symbols that have an associated arity representing how many child node it can have. Q_f is a set of accepting states. δ is a transition function of the form: $f(q_1, \ldots, q_n) \to q$, where $f \in \mathbb{F}, n = arity(f), q, q_1, \ldots, q_n \in Q$. The rule means that if a node has symbol f, and its n child nodes are in state $q_1, q_2, \ldots, q_n$ respectively, then the node will have state q. The automaton accept if there is a run in which the root of the tree is in an accepting state.*

For example, we can construct a tree automata that only accept true Boolean algebra expression. $\mathbb{F} = \{0, 1, not, and, or\}$. $Q = \{q_0, q_1\}$, $Q_f = \{q_1\}$. The rules of δ will be somethings like:

$$
\begin{gathered}
0 \to q_0 \\
1 \to q_1 \\
not(q_0) \to q_1 \\
not(q_1) \to q_0 \\
and(q_0, q_0) \to q_0 \\
and(a_0, q_1) \to q_0 \\
\ldots
\end{gathered}
$$

An expression like $(0\ or\ 1)\ and\ (1\ and\ (0\ or\ 1))$ will be parsed and recognised by the following tree:

$$
\begin{tikzcd}
& & \text{and} \arrow[dl] \arrow[drr] & & & & \\
& \text{or} \arrow[dl] \arrow[dr] & & & \text{and} \arrow[dl] \arrow[dr] & & \\
0 & & 1 & 1 & & \text{or} \arrow[dl] \arrow[dr] & \\
& & & & 0 & & 1
\end{tikzcd}
$$

If we start from the leaves and using the rules above, the root node and will have a state q_1, so the tree will be accepted by the tree automaton.

A. Cayley Presentations of $\mathbb{Z}_2 \wr \mathbb{Z}^2$ using Tree Automata

We can use the same ideas as the transducer to compute constant multiplication, with some tweaking, to construct a tree automaton for constant multiplication of 2D lamplighter group $\mathbb{Z}_2 \wr \mathbb{Z}^2$. First, we describe a tree-regular language to represent the group's elements:

Supposed that the destination marker is at (a, b), then the root of the tree is at (0, b), that is, same column as destination marker, and same row as the origin. The $root$ symbol has arity 4, that is, 4 branches which represents going up, down, left, right:

- The up branch is nil or starts with the symbol up with arity 3, that is, 3 sub-branches, representing going up, left and right
  - The up sub-branch can be either another up branch.
  - left/right sub-branch is a list of the basic symbols with arity 1: $0, 1, \bullet, *$ (they have the same meaning as before, $\bullet$ marks the origin, $*$ marks the destination), and ends with nil (arity 0) for termination
- The down branch is nil or starts with the symbol $down$ with arity 3, that is, 3 sub-branches, representing going down, left and right. Further details is similar to the up branch.
- The left/right branch is similar to the left/right sub-branch of the up branch described above

Rules about which elements and which rows to be represented is also the same as the transducer representation.

For example, the element mentioned earlier, the lamps are switched at (0, -1), (2, 4), (3, 0), (3, 3) and the lamplighter stops at (2, 1), will be represented by the following tree:

$$
\begin{tikzcd}
& & & & \text{nil} & & & & & & \\
\text{nil} & 1 \arrow[l] & \bullet \arrow[l] & 0 \arrow[l] & \text{root} \arrow[l] \arrow[r] \arrow[u] \arrow[d] & \text{nil} & & & & & \\
& & & \text{nil} & \text{down} \arrow[l] \arrow[r] \arrow[d] & \text{nil} & & & & & \\
& & & \text{nil} & \text{down} \arrow[l] \arrow[r] \arrow[d] & * \arrow[r] & 0 \arrow[r] & 0 \arrow[r] & 0 \arrow[r] & 1 \arrow[r] & \text{nil} \\
& & \text{nil} & 1 \arrow[l] & \text{down} \arrow[l] \arrow[r] \arrow[d] & 0 \arrow[r] & 0 \arrow[r] & 1 \arrow[r] & \text{nil} & & \\
& & & & \text{nil} & & & & & &
\end{tikzcd}
$$

Theorem 21. *Let G be a finite group. Then $G \wr \mathbb{Z}^2$ is Cayley tree automata presented.*

Proof. Again, we need to show that multiplication with the generators: switching current lamp, moving unit distance in up, down, left, right directions are tree-automatic.

Switching current lamp can be done by only accepting convolution of the same symbols, except for current position, where an element $g \in G$ is multiplied with a corresponding generator $s \in S$, where S is a generating set of G. This operation can be coded into a tree automaton due to finiteness of G.

Moving up / down can be done by removing the $*$ symbol in the row where it appears and adding it in the row above / below. This row can be guessed using non-determinism.

Moving left / right can be done by shifting every row to the right / left. The tree-automaton uses states to remember the previous cell of the row in the input tree and write that to the current position of the output tree. For example, a move left in the example tree will result in the following tree:

$$\begin{array}{l}
\qquad\qquad\quad \text{nil} \\
\qquad\qquad\quad \uparrow \\
\text{nil} \longleftarrow 1 \longleftarrow \text{root} \longrightarrow \bullet \longrightarrow 0 \longrightarrow \text{nil} \\
\qquad\qquad\quad \downarrow \\
\qquad \text{nil} \longleftarrow \text{down} \longrightarrow \text{nil} \\
\qquad\qquad\quad \downarrow \\
\qquad \text{nil} \longleftarrow \text{down} \longrightarrow * \longrightarrow 0 \longrightarrow 0 \longrightarrow 0 \longrightarrow 0 \longrightarrow 1 \longrightarrow \text{nil} \\
\qquad\qquad\quad \downarrow \\
\qquad \text{nil} \longleftarrow \text{down} \longrightarrow 1 \longrightarrow 0 \longrightarrow 0 \longrightarrow 1 \longrightarrow \text{nil} \\
\qquad\qquad\quad \downarrow \\
\qquad\qquad\quad \text{nil}
\end{array}$$

This completes the proof. □

This technique of shifting the rows is, however, only applicable to the 2 dimensional case. For higher dimensions, we would have to shift a 2D planes and it is not clear whether this can be done in a tree structure. Thus, for tree automata, we have a weaker result that is only true for the 2 dimensional case.

B. Lamplighters over Free Groups

Having studied the lamplighter groups over $\mathbb{Z}^n$, we have seen their interesting properties with respect to transducers and tree automata. In this subsection, we study the lamplighter groups over free groups, $\mathbb{Z}_k \wr F_n$, that is, the wreath product of the finite cyclic group over the free group of n generators. Berdinsky [5] showed that it is Cayley pushdown automatic. Here we show that it is also Cayley tree automatic.

First, we give an explicit construction for the group $G \wr F_n$, where G is a finite group. Later we slightly generalise the construction for any Cayley tree automata presented group G. Recall that each element of $G \wr F_n$ can be considered as a pair (f, b) where $f : F_n \to G$ is a mapping with finite support (that is, finitely many elements of F_n is mapped to non-neutral elements of $\mathbb{Z}_k$) and $b \in F_n$. Given two such pairs (f, b) and (f', b'), the group multiplication is performed as $(f, b)(f', b') := (f \circ bf', b \circ b')$, where $bf'(x) = f'(b^{-1}x)$ for any $x \in F_n$.

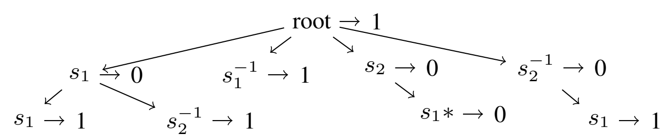
Theorem 22. *Given any finite group G, the group $G \wr F_n$ is Cayley tree automata presented for any $n \geq 1$.*

Proof. We wish to construct a Cayley tree automata presentation for $G \wr F_n$. Let $s_1, s_2, \ldots, s_n$ and their inverses denote the generators of F_n. We observe that since f has finite support, it can be finitely represented by tabulating the mapping of all the string $x \in \{s_1, s_2, \ldots, s_n, s_1^{-1}, s_2^{-1}, \ldots, s_n^{-1}\}^*$ such that $f(x)$ is not the neutral element. We can collect all these strings to form a tree as followed:

- The root node has the value of $f(\epsilon)$ as one branch, and $2n$ other branches, each representing having $s_1, s_2, \ldots, s_n, s_1^{-1}, s_2^{-1}, \ldots, s_n^{-1}$ as the first symbol in the string.
- Each branch has $2n$ sub-branches, $(2n - 1)$ of which represent multiplying with one of the generators other than the inverse of the current branch (that is, if current branch is s_k, then there will be no s_k^{-1} sub-branch). The last branch is the value of f at the current position.
- The tree only extends to the elements in F_n that is mapped by f to non-neutral elements.

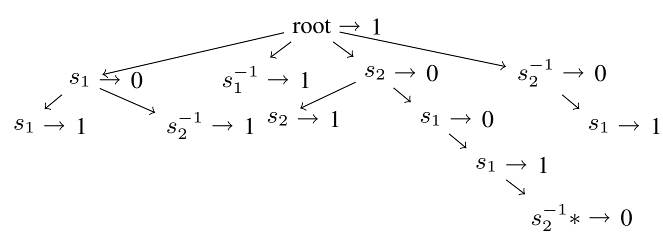
To represent (f, b) in $G \wr F_n$, we follow the tree of f using the string $b \in F_n$ and put a $*$ on the resulting node.

For example, for $f \in \mathbb{Z}_2 \wr F_2$ with $f(\epsilon) = f(s_1^2) = f(s_1 s_2^{-1}) = f(s_1^{-1}) = f(s_2^{-1} s_1) = 1$ and $b = s_2 s_1$ will be represented by the following tree:



Now we need to modify the tree to represent the result when multiplying with a constant (f', b'). bb' is easy to compute, the tree automata just need to follow the concatenated string bb', making extra branches if needed, and mark the node at the end with $*$. To compute $f \circ bf'$, we first observe that for each string x' such that $f'(x')$ is not the neutral element, the value of $f \circ bf'$ will be affected at x where $bf'(x) = f'(x')$. This implies $b^{-1}x = x'$ or $x = bx'$. Hence, we can compute $f \circ bf'$ from f by modifying finitely many nodes that represent strings of the form bx' where x' is in the finite support of f' (and thus are constants). These nodes can all be reached by a constant path from the node representing b, which is marked, so the tree automata can find these nodes and multiply the value in G with $f'(x')$.

Continuing from the previous example, if we multiply (f, b) with (f', b') where $f'(s_1^{-1} s_2) = f'(s_1) = 1$, $b' = s_1 s_2^{-1}$, we will get the following tree for $(f, b)(f', b')$:



The affected paths are $s_2 s_2$ and $s_2 s_1 s_1$. □

The above construction can be generalized so that the group G does not have to be finite, but Cayley transducer presented. The construction is very similar to the above, with the exception that we glue tree presentations to the corresponding nodes.

Theorem 23. *If G is a Cayley tree automatic group and $n \geq 1$ then also $G \wr F_n$ is Cayley tree automatic, where F_n is the free group with n generators.*

Proof. Let $\phi : D \to G$ be a corresponding Cayley tree automatic presentation of the group G. To show that $G \wr F_n$ is Cayley tree automatic group, one needs to find a corresponding presentation $\psi : N \to G \wr F_n$. N is constructed such that one glues a representative tree for an element $g \in G$ in positions in Cayley graph of F_n as a subtree. Here each element $g \in G \wr F_n$ can be viewed as a pair (k, y), where $k \in K = \bigoplus_{F_n} G$ and $y \in F_n$ indicates the position of lamplighter. Group multiplication of such pairs has the following form:

$$(k_1, y_1) \circ (k_2, y_2) = (k_1 \circ_K y_1 k_2, y_1 \circ_{F_n} y_2).$$

A presentation of an element $(k, y) \in G \wr F_n$ can be obtained in the following fashion. We start with a Cayley graph $\Gamma(F_n)$ of F_n, including a node $x \in F_n$ and all of its prefixes only if $k(x) \neq id_G$ or $x = y$. Let resulting tree be denoted as T'. For each node $x \in T'$ such that $k(x) \neq id_G$, a corresponding subtree $\phi^{-1}(k(x)) \in D$ is glued to the node x. Let T denote the resulting tree. Observe that if r is arity of trees in D, then arity of resulting tree T is $r + 2n + 1$. Furthermore, if the collection of labels corresponding to trees in D is also extended by addition of two new labels corresponding to nodes in Cayley graph $\Gamma(F_n)$ and label indicating a position of lamplighter y. Let us denote two new labels as f and p respectively. In this way, we described a procedure for derivation of presentations for an arbitrary element $(k, y) \in G \wr F_n$. Let N denote the collection of such presentation and ψ a mapping between presentations and group elements. We claim that $\psi : N \to G \wr F_n$ is Cayley tree automata group presentation. In order to verify this claim we need to check two conditions: tree automata recognisability of N and tree automaticity of functions $x \to x \circ t$ for each generator t of $G \wr F_n$. Observe that presentations in N are obtained by gluing tree presentations of G onto $\Gamma(F_n)$, hence tree automaticity of N readily follows. As for multiplication by generators, recall that we are dealing with two type of generators corresponding to either change of state of current lamp or change of lamplighter's position. More precisely, let $S = \{s_i\}_{i=1}^m$ be a generating set for G and $\{e_j\}_{j=1}^n$ be a generating set for F_n. Then $G \wr F_n$ is generated by $\{(k_i, id_{F_n})\} \bigcup \{(id_K, e_j)\}$, where $k_i \in K$ is defined as follows:

$$k_i(x) = \begin{cases} s_i, & \text{if } x = id_{F_n}; \\ id_G, & \text{otherwise.} \end{cases}$$

Let us show tree automaticity of a map $x \to x \circ t$ for each type of generator t.

Changing Lamp's State. Let a generator t be of the form (k_i, id_{F_n}). Given an element $x = (k, y) \in G \wr F_n$, the multiplication $x \to x \circ t$ can be written as

$$(k, y) \to (k \circ_K y k_i, y).$$

In terms of tree presentations, this function corresponds to replacement of the element k located at node y with the element $k \circ_K k_i$. Since current presentation is based on Cayley tree automatic presentation $\phi : D \to G$, it follows that the underlying function is tree automatic.

Shift of the Lamplighter. Let a generator t be of the form (id_K, e_j). Given an element $x = (k, y) \in G \wr F_n$, the multiplication $x \to x \circ t$ can be written as

$$(k, y) \to (k, y \circ_{F_n} e_j).$$

In terms of tree presentation, this function corresponds to the shift of the lamplighter from node y to the node $y \circ_{F_n} e_j$. Since shifts of a single label by a finite distance are tree automatic, tree automaticity of above function follows. □

V. Tree Automata and Pushdown Automata

Berdinsky [5] showed that the group studied in the previous section, $\mathbb{Z}_k \wr F_n$, has Cayley graphs that can be recognised by deterministic pushdown automata. As we have shown that they are also recognised by tree automata, we get an alternative proof by providing a link between tree automata and pushdown automata in general. For this, we use *asynchronous pushdown automata* which extend the traditional pushdown automata in the same way as how transducers extend finite-state automata. Similar to a conventional pushdown automata, our asynchronous pushdown automata has finite number of states, a stack, and tapes. Transitions from one state to another are based on the symbol at the top of the stack as well as the current symbols in the tapes. Acceptance is by the final state of the pushdown automata. The difference of our automata is that it will have multiple tapes, one for each input/result of the function. At anytime, the automata can read one symbol from one tape, that is, it can read different tapes at different speed (hence the name asynchronous).

Theorem 24. *Any function that can be computed by tree automata can also be computed by deterministic asynchronous pushdown automata.*

Proof. For simplicity, we will assume that the tree in the tree automata is a binary tree. Any tree automata with a finitely branching tree can be transformed into a tree automata with a binary tree. For example, a single $node$ with n children $node_1, node_2, \ldots, node_n$ can be transformed into a binary tree by adding new nodes:

$$\begin{tikzcd}
& node \arrow[dl] \arrow[dr] & & & \\
node_1 & & new_1 \arrow[dl] \arrow[dr] & & \\
& node_2 & & \cdots \arrow[dr] & \\
& & & & new_{n-1} \arrow[dl] \\
& & & node_n &
\end{tikzcd}$$

The old transition rules of the form $node(q_1, q_2, \ldots, q_n) \to q$ can be rewritten as several rules:

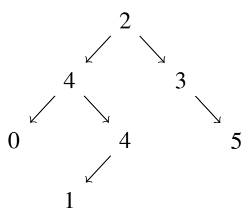
$$new_{n-1}(q_n) \to (q_n)$$
$$new_{n-2}(q_{n-1}, (q_n)) \to (q_{n-1}, q_n)$$
$$\ldots$$
$$node(q_1, (q_2, q_3, \ldots, q_n)) \to q$$

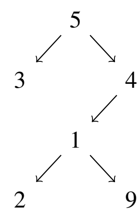
where (q_{n-1}, q_n), $(q_2, q_3, \ldots, q_n)$, etc. are newly created composite symbols. It's easy to see that this new binary tree automata will compute exactly the same function as the old one.

Next, we describe a way to linearize a binary tree by using brackets. This is not difficult. If x is a node in the binary tree, let $x.label$ denote its label, $x.left$ denote its left branch and $x.right$ denote its right branch. We define a recursive formula:

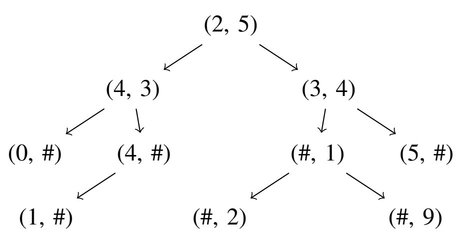
1. If x is the null node then Linearize(x) = (-);
2. If x is not the null node then Linearize(x) = (x.label Linearize(x.left) Linearize(x.right)).

For example, the two trees below are linearised to `(2(4(0(-)(-))(4(1(-)(-))(-)))(3(-)(5(-)(-))))` and to `(5(3(-)(-))(4(1(2(-)(-))(9(-)(-)))(-)))`, respectively:





It should be noted that this linearized tree is a context-free grammar, so the pushdown automata can certainly check that the input tapes follow the grammar. Hence, we omit details of this check and only describe how the asynchronous pushdown automata work to simulate the tree automata. First, recall that the tree automata compute a relation between 2 trees by overlaying the trees upon one another, a.k.a. convolution. So if the tree automata computes a relation between the above 2 trees it will run on the following overlayed tree:



where each symbol is a convolution of the symbols of the 2 trees, and # denotes that the corresponding tree does not have a node at that position. The tree automata then works from the leaves, assigning states to the leaves and then non-deterministically arrive at a state for the inner nodes based on the states of the child nodes. Eventually, the root node will be assigned a state, and if it is one of the final states, the 2 trees are accepted.

The above process can be simulated using an asynchronous pushdown automata using the linearised representations of the 2 trees as inputs. Initially, the automata read the 2 inputs at the same speed, pushing the convolutions of the symbols (or just the bracket) onto the stack. The following shows the state of the pushdown automata after reading 5 symbols from each tape. We indicate the last symbol read from each tape by adding spaces after it. The stack is written from bottom to top, that is, the last element pushed to the stack is on the right.

```
T1: (2(4(   0(-)(-))(4(1(-)(-))(-)))(3(-)(5(-)(-))))
T2: (5(3(   -)(-))(4(1(2(-)(-))(9(-)(-)))(-)))
```

$\texttt{St:}\ (\begin{bmatrix}2\\5\end{bmatrix} (\begin{bmatrix}4\\3\end{bmatrix} ($

When the automata reads a − symbol, it pauses reading from that tape, and only continue reading from the other tape. New convolutions pushed to the stack will have − as the symbol for the paused tape:

```
T1: (2(4(0(-   )(-))(4(1(-)(-))(-)))(3(-)(5(-)(-))))
T2: (5(3(-   )(-))(4(1(2(-)(-))(9(-)(-)))(-)))
```

$\texttt{St:}\ (\begin{bmatrix}2\\5\end{bmatrix} (\begin{bmatrix}4\\3\end{bmatrix} ((\begin{bmatrix}0\\-\end{bmatrix} (\begin{bmatrix}-\\-\end{bmatrix}$

When the automata reads a closing bracket `)`, it has finished reading a node. If the node is a null node, the previous 2 symbols should be (- which are popped from the stack. Also, for a null node, it will get assigned a state q_0 in the original tree automata. The pushdown automata push this state symbol onto the stack.

```
T1: (2(4(0(-)(-   )(4(1(-)(-))(-)))(3(-)(5(-)(-))))
T2:(5(3(-   )(-))(4(1(2(-)(-))(9(-)(-)))(-)))
```

$\texttt{St:}\ (\begin{bmatrix}2\\5\end{bmatrix} (\begin{bmatrix}4\\3\end{bmatrix} (\begin{bmatrix}0\\-\end{bmatrix} q_0\ q_0$

When the automata reads a closing bracket `)` and it is not a null node, then at the top of the stack will be 2 state symbols indicating the state of the 2 child nodes, as well as the label of the node. The pushdown automata pop these symbols one by one, remembering them. After the corresponding opening bracket `(`, is popped, the pushdown automata finds the state symbol that the node should have if the tree was run in a tree automata and push that state symbol onto the stack. Continuing in our example, let's say the original tree automata has the transition rule: $(0, \#)(q_0, q_0) \to q_1$, then the pushdown automata will proceed as followed:

```
T1: (2(4(0(-)(-))   (4(1(-)(-))(-)))(3(-)(5(-)(-))))
T2: (5(3(-   )(-))(4(1(2(-)(-))(9(-)(-)))(-)))
```

$\texttt{St:}\ (\begin{bmatrix}2\\5\end{bmatrix} (\begin{bmatrix}4\\3\end{bmatrix} q_1$

The pushdown automata also checks the label of the parent node (which would be $\begin{bmatrix}4\\3\end{bmatrix}$ in our current example). If it no longers contains −, the automata reads the closing bracket from the paused tape and then resumes reading from the paused tape at the same speed as the other tape.

```
T1: (2(4(0(-)(-))(4   (1(-)(-))(-)))(3(-)(5(-)(-))))
```

```
T2: (5(3(-)(-  ))(4(1(2(-)(-))(9(-)(-)))(-)))
```

St: ($\begin{bmatrix}2\\5\end{bmatrix}$ ($\begin{bmatrix}4\\3\end{bmatrix}$ q_1 ($\begin{bmatrix}4\\-\end{bmatrix}$

The process continues until the last closing brackets in all the tapes are read, after which the stack will be left with a single symbol, which is the state of the root node. The pushdown automata accepts if this state is a final state of the tree automata.

We note that in the construction of the pushdown automata from the tree automata above, we do not need non-determinism, that is, it is totally deterministic. Hence, if the tree-automata is deterministic, the asynchronous pushdown automata would also be deterministic. We also know that non-deterministic tree-automata can be readily converted to an equivalent deterministic one (the proof of this is similar to the well-known analogous fact in normal finite state automata, hence it is omitted here). Hence, it follows that any tree-automatic function can be computed by a deterministic asynchronous pushdown automata as we described above. □

This Theorem 24, together with our previous results in Theorem 21 and Theorem 22 that $G \wr \mathbb{Z}^2$ and $\mathbb{Z}_k \wr F_n$ have tree-automatic one-way constant multiplication, imply that one-way constant multiplication in these groups is also recognisable by a deterministic asynchronous pushdown automata. Here, it is worth comparing our work with earlier results proven in [4]. In that paper, the authors define the notion of poly-pushdown groups, which are groups whose one-way constant multiplication can be computed by finitely many pushdown automata running independently in parallel. They also prove that this property of poly-pushdown is preserved when taking the wreath product. From their results, we can also deduce that the groups $G \wr \mathbb{Z}^2$ and $\mathbb{Z}_k \wr F_n$ are poly-pushdown. This result is quite similar to ours, as it shows that constant multiplication in these groups can be computed by some kinds of pushdown automata.

VI. Summary

We have investigated two kind of automatic presentation of lamplighter groups: one is based on transducers, while another is based on tree automata. For each of these cases there corresponds two definitions depending on automaticity of either whole group operation or multiplication by generators. The latter presentation is called Cayley presentations in this paper. It has been shown that one-to-one transducer presentability is closed under lamplighter products over $\mathbb{Z}$. As for lamplighter product over $\mathbb{Z}^d$, we have shown that $G \wr \mathbb{Z}^d$ is Cayley transducer presented whenever G is Cayley transducer presented. In case of tree automata presentations, we have shown that $G \wr \mathbb{Z}^2$ is Cayley tree automata presented whenever G is finite. Finally, it has been shown that $G \wr F_d$ is Cayley tree automatic whenever G is. Investigations on monoids and transducers are delayed to the journal version, one upcoming result is that for ordinals $\alpha < \omega^\omega$, the monoid of ordinals $\beta < \omega^\alpha$ and ordinal addition is transducer presentable.

References

[1] Margarita Amchislavska. The geometry of generalized lamplighter groups. *PhD Thesis, Cornell University, 2014*

[2] Dana Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[3] Gilbert Baumslag. Wreath products and finitely presented groups. *Mathematische Zeitschrift* 75: 22–28, 1961.

[4] Gilbert Baumslag, Michael Shapiro and Hamish Short. Parallel poly-pushdown groups. *Journal of Pure and Applied Algebra* 140(3): 209–227, 1999.

[5] Dmitry Berdinsky. *Representations of Transitive Graphs by Automata.* PhD Thesis, University of Auckland, 2017.

[6] Dmitry Berdinsky and Bakhadyr Khoussainov. Cayley automatic representations of wreath products. *International Journal of Foundations of Computer Science*, 27(2):147–159, 2016.

[7] Jean Berstel. *Transductions and Context-free Languages*. Teubner, 1979. http://www-igm.univ-mlv.fr/˜berstel/LivreTransductions/LivreTransductions.html

[8] Achim Blumensath. *Automatic structures*. Diploma thesis, Department of Computer Science, RWTH Aachen, 1999.

[9] Achim Blumensath and Erich Grädel. Automatic structures. *Fifteenth Annual IEEE Symposium on Logic in Computer Science*, LICS 2000, pages 51–62, 2000.

[10] John Case, Sanjay Jain, Samuel Seah and Frank Stephan. Automatic functions, linear time and learning. *Logical Methods in Computer Science*, 9(3), 2013.

[11] Calvin C. Elgot and Jorge E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68, 1965.

[12] David B.A. Epstein, James W. Cannon, Derek F. Holt, Silvio V.F. Levy, Micheal S. Paterson and William P. Thurston. *Word Processing in Groups.* Jones and Bartlett Publishers, Boston, 1992.

[13] Rostislav I. Grigorchuk and Andrzej Żuk. The Lamplighter group as a group generated by a 2-state automaton and its spectrum. *Geometriae Dedicata*, 87(1–3):209–244, 2001.

[14] Bernard R. Hodgson. *Théories décidables par automate fini*. Ph.D. thesis, Département de mathématiques et de statistique, Université de Montréal, 1976.

[15] Bernard R. Hodgson. Décidabilité par automate fini. *Annales des sciences mathématiques du Québec*, 7(1):39–57, 1983.

[16] Sanjay Jain, Bakhadyr Khoussainov and Frank Stephan. Finitely generated semiautomatic groups. *Computability*, 7(2-3):273–287, 2018.

[17] Sanjay Jain, Bakhadyr Khoussainov, Frank Stephan, Dan Teng and Siyuan Zou. Semiautomatic structures. *Theory of Computing Systems*, 61(4):1254–1287, 2017.

[18] Olga Kharlampovich, Bakhadyr Khoussainov and Alexei Miasnikov. From automatic structures to automatic groups. *Groups, Geometry and Dynamics*, 8(1):157–198, 2014.

[19] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. *Logical and Computational Complexity*, (International Workshop LCC 1994). Springer LNCS 960:367–392, 1995.

[20] Birzhan Moldagaliyev. *A studay of Randomness and Groups Using Finite Automata.* PhD Thesis, National University of Singapore, 2018.

[21] Maria Monks. *Closure properties of predicates recognized by deterministic and non-deterministic asynchronous automata.* Technical Report on http://www.arxiv.org/abs/1010.3039.

[22] Graham P. Oliver and Richard M. Thomas. Finitely generated groups with automatic presentations. *STACS 2005, 693-704, Lecture Notes in Computer Science, 3404, Springer, Berlin, 2005.*

[23] Sasha Rubin. Automata presenting structures: a survey of the finite string case. *The Bulletin of Symbolic Logic*, 14:169–209, 2008.

[24] Takeshi Shinohara. Polynomial time inference of extended regular pattern languages. *RIMS Symposia on Software Science and Engineering*, Kyoto, Japan, Proceedings. *Springer LNCS*, 147:115–127, 1982.

[25] Tien Dat Tran. *Automata Theory and Groups.* Final Year Project Report, National University of Singapore, 2018