



UNICAMP

UNIVERSIDADE ESTADUAL DE
CAMPINAS

Instituto de Matemática, Estatística e
Computação Científica

BIANCA BOEIRA DORNELAS

Co-Context-Free Groups

Grupos Co-Livres de Contexto

Campinas

2019

Bianca Boeira Dornelas

Co-Context-Free Groups

Grupos Co-Livres de Contexto

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestra em Matemática.

Dissertation presented to the Institute of Mathematics, Statistics and Scientific Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Mathematics.

Supervisora: Dessislava Hristova Kochloukova

Co-supervisor: Francesco Matucci

Este exemplar corresponde à versão final da Dissertação defendida pela aluna Bianca Boeira Dornelas e orientada pela Profa. Dra. Dessislava Hristova Kochloukova.

Campinas

2019

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

D735c Dornelas, Bianca Boeira, 1996-
Co-context-free groups / Bianca Boeira Dornelas. – Campinas, SP : [s.n.],
2019.

Orientador: Dessislava Hristova Kochloukova.

Coorientador: Francesco Matucci.

Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de
Matemática, Estatística e Computação Científica.

1. Teoria dos grupos. 2. Thompson, Grupos de. 3. Autômatos a estados
finitos. 4. Linguagens livres de contexto. 5. Grupos co-livres de contexto. I.
Kochloukova, Dessislava Hristova, 1970-. II. Matucci, Francesco, 1977-. III.
Universidade Estadual de Campinas. Instituto de Matemática, Estatística e
Computação Científica. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Grupos co-livres de contexto

Palavras-chave em inglês:

Group theory

Thompson groups

Finite state automata

Context-free languages

Co-context-free groups

Área de concentração: Matemática

Titulação: Mestra em Matemática

Banca examinadora:

Dessislava Hristova Kochloukova [Orientador]

Adriano Adrega de Moura

Alex Carrazedo Dantas

Data de defesa: 23-09-2019

Programa de Pós-Graduação: Matemática

Identificação e informações acadêmicas do(a) aluno(a)

- ORCID do autor: <https://orcid.org/0000-0002-4827-4663>

- Currículo Lattes do autor: <http://lattes.cnpq.br/5087073027808281>

**Dissertação de Mestrado defendida em 23 de setembro de 2019 e aprovada
pela banca examinadora composta pelos Profs. Drs.**

Prof(a). Dr(a). DESSISLAVA HRISTOVA KOCHLOUKOVA

Prof(a). Dr(a). ALEX CARRAZEDO DANTAS

Prof(a). Dr(a). ADRIANO ADREGA DE MOURA

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

To my father Manoel Dornelas de Souza, to my mother Rita Carla Boeira and to my brother Eric Boeira Dornelas. You are the ones responsible for the beginning of my love for science.

Acknowledgements

First of all, I would like to thank my friends for all support during this dissertation. It would not have been possible to finish this work without them. I am grateful to Matheus Manzatto and Mateus Sangalli, for sharing experiences and thoughts about the process of writing a dissertation. I am deeply grateful to Raissa Nouer for all her love and support during these years. I am grateful to Marcelo Watanabe, Roberto Zurita, Danilo Kanno, Lucas Campos, Davi de Alvarenga, Flávio Kajiwara, Fábio Meneghetti and Aline D'Oliveira for all our random conversations, I would not have even a bachelor degree if not for them. The same holds for everyone in the Unicamp Taekwondo Team, our trainings were always the best place to let the stress melt away. Also, Altair Oliveira and Marcelo Miranda were life savers when it came to dealing with our advisor departure and they always gave me nice tips for life at IMECC. Laura Meirelles, Giovanna Bombonati, Maria Carolina Lourenço and Naiara Godoi helped me in so many difficult decisions, I cannot ever thank them enough. There is a huge list of friends names I would still like to include here, but I will limit myself to identify only the groups where everyone is important to me and helped me in some way or another to get through my undergraduate and graduate studies, and they will know it: all my dear miadas, my dear friends of big 3rd and everyone in the abelian group.

I would like to express my gratitude to my advisor Francesco Matucci for introducing to me this area of Mathematics. I remember the first time he said something to me about Galois Theory and got me enthusiastic about it. Ever since, it has been a constant pleasure to work with him and I cannot express properly all my gratitude for his constant support and patience while he directed me through these years and helped me with many important decisions for my future career. It has been a great honor to work with him and I could not be more happy about it.

I would like to thank as well professor Dessislava Kochloukova for being my formal advisor in the last months before the graduation, for her great tips on how to deal with bureaucracies and her support on absolutely any matter where we needed her.

I also want to thank Yury Popov for his help with an article which I could only find in Russian.

I want to thank the faculty and staff of the IMECC, specially from the Department of Mathematics and from the Secretary of Graduate Studies, they saved me lots of trouble many times.

This study was financed by Fundação de Amparo à Pesquisa do Estado de São Paulo,

FAPESP, in part through the process 2017/24373-1, from 03/01/2018 to 11/30/2018, and in part through the process 2018/24172-9, from 03/01/2019 to 09/30/2019. I thank FAPESP and the people of São Paulo for that financial support.

Resumo

Na presente dissertação de mestrado, estudamos grupos do ponto de vista da ciência da computação. O primeiro objetivo é entender a construção da Teoria de Chomsky em Grupos através da construção dos teoremas de Anisimov [21] e de Muller & Schupp [10]. Depois disso, são estudadas propriedades gerais da próxima classe de grupos na Hierarquia, os grupos co-livres de contexto, através de [15]. Finalizamos apresentando os grupos F , T e V de Thompson e demonstrando, com as técnicas de [18], que V é um grupo co-livre de contexto.

Palavras-Chave: Grupos de Thompson. Grupos que agem sobre árvores. Autômatos a estados finitos. Hierarquia de grupos através de autômatos. Linguagens livres de contexto.

Abstract

In the present work we study groups from the point of view of scientific computation. The first goal is to understand the construction of the Chomsky's Hierarchy for Groups by constructing Anisimov's [21] and Muller & Schupp's [10] theorems. After that, general properties of the next class of groups in the Hierarchy, the co-context-free groups, are studied using [15]. We end presenting the Thompson's groups F, T and V and showing, with the techniques from [18], that V is a co-context-free group.

Keywords: Thompson's groups. Groups acting on trees. Finite state automata. Groups hierarchy through automata. Context-free languages.

Contents

Introduction	11
1 Preliminaries	14
1.1 General Group Theory	14
1.2 Graphs and Free Groups	17
1.3 Group Presentations	22
1.4 Languages and Dehn's Problems	23
2 Anisimov's Theorem	27
2.1 Finite State Automata	27
2.2 Main Theorem	31
3 Muller and Schupp's Theorem	33
3.1 Pushdown Automata	33
3.1.1 Context-Free Languages	33
3.1.2 One-Counter Languages	50
3.2 The Theorem of Muller and Schupp	51
3.2.1 Some Definitions	51
3.2.2 Main Results	54
4 Co-Context-Free Groups	56
4.1 Finitely Generated Subgroups	56
4.2 Finite Direct Products	59
4.3 Finite Index Overgroups	66
4.4 Wreath Products	68
4.5 Other Properties	71
5 Thompson Groups	74
5.1 Thompson Groups	74
5.2 Higman-Thompson Groups	77
5.3 Houghton Groups	81
6 Lehnert's Conjecture	85
Final Remarks	94
 BIBLIOGRAPHY	 95

Introduction

Our main goal in this dissertation is the study of the classification of groups according to some formal language properties of its Language of the Word Problem. In particular, we are interested in the study of co-context-free groups. For a group G generated by a finite set S , one can consider the set $\text{WP}(G, S)$ of all the words in the free group with generators in S , $\mathbb{F}(S)$, which are equal to the identity as elements of G . This set is known as the *Language of the Word Problem of G* (with respect to S).

It is possible to study properties of $\text{WP}(G, S)$ and obtain results about G , as Anisimov showed when he completely classified the class of finite groups with respect to languages.

Theorem A (Anisimov, [1]). *A group G generated by a finite set S is finite if, and only if, its Language of the Word Problem $\text{WP}(G, S)$ is a regular language.*

The property of $\text{WP}(G, S)$ being a regular language is studied by means of automata. A *finite state automaton* is a graph with a finite number of vertices, called *states*, and directed edges, called *transitions*, which, together with some more rules and components, models a machine/computer. A very important component is the alphabet X with whose letters the transitions are labelled. This guarantees that the automaton accepts words of the free group $\mathbb{F}(X)$, where accepting a word means that one can follow a path inside the graph, obeying the rules given by the automaton, forming that same word.

If a language is accepted by a finite state automaton, then the language is regular. There are other types of formal languages and there exists a hierarchy classifying them, known as Chomsky's Hierarchy.

Definition B (Chomsky's Hierarchy, [9]). *Chomsky's Hierarchy classifies formal languages in four classes, each of them containing the next one.*

- Type 0: *Recursively countable languages. (Accepted by Turing machines.)*
- Type 1: *Context-sensitive languages. (Accepted by linearly nondeterministic Turing machines.)*
- Type 2: *Context-free languages. (Accepted by nondeterministic pushdown automata.)*
- Type 3: *Regular languages. (Accepted by deterministic finite state automata.)*

For any language, to know properties of a generating grammar is the same as having information about the language structure, however one needs to first discover a generating

grammar and the question about how to choose such a grammar could not be answered, in Chomsky's view, before developing a general theory of the linguistic structure and the formal properties of grammars. Although we present the full hierarchy, in respect of Chomsky's work, our work does not deal with the first two types of languages.

Anisimov's theorem connects type 3 languages to finite groups, leading to the question whether there is a relation between Chomsky's Hierarchy for languages and a possible hierarchy for finitely generated groups. Some years after the publication of Anisimov's result, Muller & Schupp's result contributed constructing a further step in such a structure.

Theorem C (Muller, Schupp, [23, 22]). *A group G generated by a finite set S is virtually free if, and only if, its Language of the Word Problem $WP(G, S)$ is a context-free language.*

Context-free languages are the ones accepted by a *pushdown automaton*. These automata are a generalization of the finite state ones. As an intermediate step of the hierarchy for groups, it is possible to define one-counter languages, leading to the result by Herbst:

Theorem D (Herbst, [12]). *A group G generated by a finite set S is virtually cyclic if, and only if, its Language of the Word Problem $WP(G, S)$ is a one-counter language.*

In [15], Derek, Rees, Röver and Thomas defined a new class of groups defined by automata, the *co-context-free groups*, which is currently being regarded as the next class to be classified. We will see many closure properties of this class of groups, such as closure by taking finitely generated subgroups or closure by taking direct products.

It is still an open question to completely classify co-context-free groups, as was done to the finite and the virtually cyclic groups. However, the following equivalent statement of a conjecture by Lehnert aims to give such classification [17]:

Conjecture E (Lehnert, [17]). *A group G generated by a finite set S is a subgroup of Thompson's group V if, and only if, its Language of the Co-word Problem, $coWP(G, S)$, is a co-context-free language.*

We present the basic definitions and properties of geometric group theory that will be required from the reader in Chapter 1.

Chapter 2 introduces automata theory, with the definition of finite state automata and the statement of Anisimov's result.

The pushdown and one-counter automata are studied in Chapter 3, where we also state the results leading to Muller & Schupp's theorem.

It remains to define co-context-free groups, studying their properties and proving that V is co-context-free. We divide these topics as follows: In Chapter 4 we deal with

co-context-free groups in general, passing through closure properties and seeing examples of groups which are not co-context-free. In Chapter 5, we briefly introduce Thompson groups F and T , giving more details about the group V . Their definitions and basic properties are presented. After that, in Chapter 6, we follow [18] and show that V is a co-context-free group, understanding the motivation to Lehnert's conjecture.

1. Preliminaries

In this chapter, we present some preliminary definitions and results concerning geometric group theory. The reader can consult one of the references [14, 19, 21] or any other introductory book on geometric group theory for more details.

1.1 General Group Theory

The next definitions are standard and can be found in any group theory (for example, see [25]).

Definition 1.1.1. Given two groups, (G, \cdot) and (H, \star) , a *group homomorphism* from (G, \cdot) to (H, \star) is a function

$$\phi : G \rightarrow H$$

such that for all $g, h \in G$, $\phi(g \cdot h) = \phi(g) \star \phi(h)$.

We usually refer to a given group by its underlying set only, instead of writing the pair *set+operation*.

Definition 1.1.2. Let G be a group and $H \subseteq G$ a subgroup. Given $g \in G$, the *left coset* of g with respect to H is

$$gH = \{gh \mid h \in H\}.$$

An analogous definition can be made for a right coset Hg .

Some well known facts of group theory are that two left cosets are either disjoint or the same, thus the set of all left cosets is a partition of a group G ; the left cosets are the equivalence classes for the equivalence relation $x \sim_H y$ if and only if $x = yh$ for some $h \in H$; and that one can use the axiom of choice and choose representatives for each left coset of H .

Definition 1.1.3. Given a group G and a subgroup $H \subseteq G$, a *left transversal* to H in G is a set T with one representative, exactly, for each left coset of H . That is, T is such that

$$G = \bigsqcup_{t \in T} tH$$

and therefore any element $g \in G$ can be written as $g = th$ for some $h \in H$. An analogous definition can be made for a right transversal.

Henceforth, we use only *transversal* and *coset* to refer to the left transversals and the left cosets, unless specified otherwise. It is clear that the cardinality of a transversal T is the number of left cosets of H in G . That is the exact definition of index.

Definition 1.1.4. Let G be a group and $H \subseteq G$ be a subgroup of G . Let T be a transversal for H in G . The index of H in G is

$$|G : H| = |T|$$

that is, the index of H in G is the cardinality of the collection of cosets of H in G .

Although there are several transversal sets for the same subgroup H , the index is well defined because the cardinality of all the transversal sets are the same.

Definition 1.1.5. Given G a group and a subgroup $H \subseteq G$, H is said to be *normal* (in G) if it satisfies one of the following equivalent statements:

1. $gH = Hg$ for all $g \in G$.
2. $g^{-1}Hg = H$ for all $g \in G$.
3. $g^{-1}hg \in H$ for all $g \in G, h \in H$.

Definition 1.1.6. Given N a normal subgroup of G a group, the *quotient group* (of G by N) is the set of all left cosets,

$$G/N = \{gN \mid g \in G\}$$

together with the operation

$$gN \cdot hN = (gh)N.$$

We also need the definition of *rank* of a group.

Definition 1.1.7. Let G be a group. The *rank* of G is

$$\text{rank}(G) = \min\{|S| : S \text{ generates } G\}.$$

We now give definitions related to the action of a group over some set.

Definition 1.1.8. Consider a non-empty set X . The *symmetric group* over X is

$$(\text{Sym}(X), \cdot),$$

where

$$\text{Sym}(X) = \{\phi : X \longrightarrow X \mid \phi \text{ is bijective}\}$$

and

$$\begin{aligned} \cdot : \text{Sym}(X) \times \text{Sym}(X) &\longrightarrow \text{Sym}(X) \\ (\phi, \varphi) &\mapsto \phi \circ \varphi. \end{aligned}$$

Definition 1.1.9. Consider a group G and a non-empty set X . A *left action* of G on X is a map

$$\varphi : G \times X \longrightarrow X$$

such that

1. $e \cdot x = x$ for all $x \in X$; and
2. $(gh) \cdot x = g \cdot (h \cdot x)$ for all $g, h \in G$ and $x \in X$.

Equivalently, one can consider a group homomorphism from G to the group of symmetries of X ,

$$\phi : G \longrightarrow \text{Sym}(X).$$

ϕ is called a *representation* of G .

We denote the action φ (the representation ϕ) of G on X by $G \curvearrowright_{\varphi} X$ ($G \curvearrowright_{\phi} X$), although often dropping φ (ϕ) from the notation when the function is clear from the context. Right actions can be analogously defined.

Definition 1.1.10. Given a group G acting on a non-empty set X with representation ϕ , the representation is *faithful* if the map is injective.

Definition 1.1.11. Given a group G acting on a non-empty set X and $x \in X$, the *stabilizer* of x is the subgroup (of G)

$$\text{Stab}(x) = \{g \in G \mid g \cdot x = x\}.$$

Definition 1.1.12. Let $G \curvearrowright X$. If there is $x \in X$ such that $\text{Stab}(x) = \{e\}$, then x is said to be *moved freely* by the action of G . The action is a *free action* if every $x \in X$ has $\text{Stab}(x) = \{e\}$.

Definition 1.1.13. Let $G \curvearrowright X$. The *orbit* of $x \in X$ is the set

$$\text{Orb}(x) = \{g \cdot x \mid g \in G\}.$$

Definition 1.1.14. Consider a group G . The *center* of G is the set

$$Z(G) = \{z \in G \mid zg = gz, \forall g \in G\},$$

which is always a normal subgroup of G .

Clearly, a group is abelian only if its center is the entire group.

1.2 Graphs and Free Groups

In this section we introduce basic definitions of graph theory and the theory of free groups, which are needed for this work. All the results discussed below were studied following [21].

Definition 1.2.1. A *graph* Γ is a set of vertices, $V(\Gamma)$, together with a set of edges, $E(\Gamma)$. Each edge e is associated to a non-ordered pair of vertices by the map $\text{Ends}(e) = \{v, w\}$, $v, w \in V(\Gamma)$. The two vertices v and w are called the *ends of the edge*. Two vertices, $v, w \in V(\Gamma)$, are said to be *adjacent* if there exists at least one edge $e \in E(\Gamma)$ such that $\text{Ends}(e) = \{v, w\}$.

Notice in the above definition that there could exist multiple edges between the same pair of vertices. One can picture a graph as a set of points representing the vertices and traces representing the edges. Their importance for this project comes from their connection with the free groups, which we will see shortly, and their role as automata, which will be fully presented in Chapter 2.

We now define some qualities that a graph can have.

Definition 1.2.2. A *path* (of edges) in a graph is a sequence that alternates vertices and edges, starting and ending with vertices:

$$v_0 e_1 v_1 e_2 \cdots v_{n-1} e_n v_n, \text{ where } \text{Ends}(e_i) = \{v_{i-1}, v_i\}, \forall i \in (0, n].$$

The path beginning in v_0 and ending in v_n is called a *geodesic path* between v_0 and v_n if the number of edges in the sequence is minimal.

Definition 1.2.3. A graph Γ is said to be *connected* if for any $v, w \in V(\Gamma)$, there is a path between them.

Definition 1.2.4. A *cycle*, or closed path, is a nontrivial path which begins and ends at the same vertex, v , being v the only vertex to repeat itself in the path. A *loop* is an edge e which has $\text{Ends}(e) = \{v, v\}$ for some vertex v .

Definition 1.2.5. A graph is called *simple* if it does not have loops or multiple edges, that is, there are no $e_1, e_2 \in E(\Gamma)$ such that $\text{Ends}(e_1) = \{v, v\}$ or such that $\text{Ends}(e_1) = \text{Ends}(e_2)$.

Definition 1.2.6. A *tree* is a connected graph without loops and cycles.

Definition 1.2.7. A graph Γ is *directed* if each one of its edges has an ordered pair of vertices associated to them. In other words, for each edge, one can identify the initial vertex and the final one. When representing a graph with points and dashes, the directed edges are represented by arrows, out of the first entry and into the second entry of the ordered pair of vertices associated to them.

An important note on notation: many authors indicate a directed graph as a 4-uple, $\Gamma = (V, E, s, t)$. In this notation, V is the set of vertices, E is the set of edges and s and t are functions $s, t : E \rightarrow V$. For each $e \in E$, $s(e)$ is then called the *start* of e and $t(e)$ the *tail* of e . It is clear that the association as ordered pair is $(s(e), t(e))$. We use these functions later on the text, so that the reader is advised to keep them in mind.

Definition 1.2.8. A graph Γ is *labelled* if each one of its edges has a label. When representing a graph with points and dashes, the label is a symbol or a sequence of symbols near the edges (dashes).

Definition 1.2.9. The *degree* of a vertex $v \in V(\Gamma)$ is the number of times that v is an end for some edge, i.e., $\deg(v) = \#\{e \in E(\Gamma) \mid \text{Ends}(e) = \{v, w\}, v \neq w\} + 2 \cdot \#\{e \in E(\Gamma) \mid \text{Ends}(e) = \{v, v\}\}$. If all vertices in Γ have finite degree, then Γ is *locally finite*.

Definition 1.2.10. A graph is *finite* if V and E are finite.

Note that it is possible to have a locally finite infinite graph.

As our goal is to introduce the free groups, we remember the definition of finitely generated groups and afterwards we define the relationship between graphs and groups.

Definition 1.2.11. Take a group (G, \cdot) and $S \subseteq G$. If for all $g \in G$, $g = a_1 \cdot a_2 \cdot a_3 \cdots a_n$, with $a_i \in S$ or $a_i^{-1} \in S$, for any i , then S generates G . If S is finite, then G is *finitely generated*.

Definition 1.2.12. Let G be a group and $S \subseteq G$ be a finite generating set of G . Consider the graph $\Gamma_{G,S}$ whose vertices are the elements of G and, for each $g \in G$, $s \in S$, there is a directed edge from g to $g \cdot s$ labeled by s . Those are the only vertices and edges and the graph $\Gamma_{G,S}$ is called the *Cayley graph* of G with respect to S . The Cayley graph of G with respect to S is therefore a directed, locally finite, connected and labelled (by elements of S) graph.

The first connection between graphs and groups arises with Cayley's theorem about the symmetric group of a graph.

Definition 1.2.13. If Γ is a graph, we call a bijective function $f : V(\Gamma) \rightarrow V(\Gamma)$ a *symmetry* if f is such that for every v_1, v_2 adjacent vertices, we have that $f(v_1), f(v_2)$ are also adjacent, that is, f preserves adjacency. We define as $Sym(\Gamma)$ the set of all possible symmetries of Γ . $Sym(\Gamma)$ is a group when together with the operation composition of functions and it is called the *symmetric group* of Γ .

Theorem 1.2.14 (Cayley). *Every finitely generated group can be faithfully represented as $Sym(\Gamma)$ for some connected, directed and locally finite graph Γ .*

The next result shows that elements of a finitely generated group G can act on its Cayley graph.

Proposition 1.2.15. *Let G be a group with finite generating set S , then $G \cong \text{Sym}(\Gamma_{G,S})$.*

We finish our introduction to graphs with an example.

Example 1.2.16. Take S_3 , the group of all permutations of $\{1, 2, 3\}$. One generating set is $\{(12), (123)\}$. The Cayley graph of S_3 with respect to this generating set is as Figure 1.1a. Another generating set is $\{(12), (23), (13)\}$, being the correspondent Cayley graph shown in Figure 1.1b.

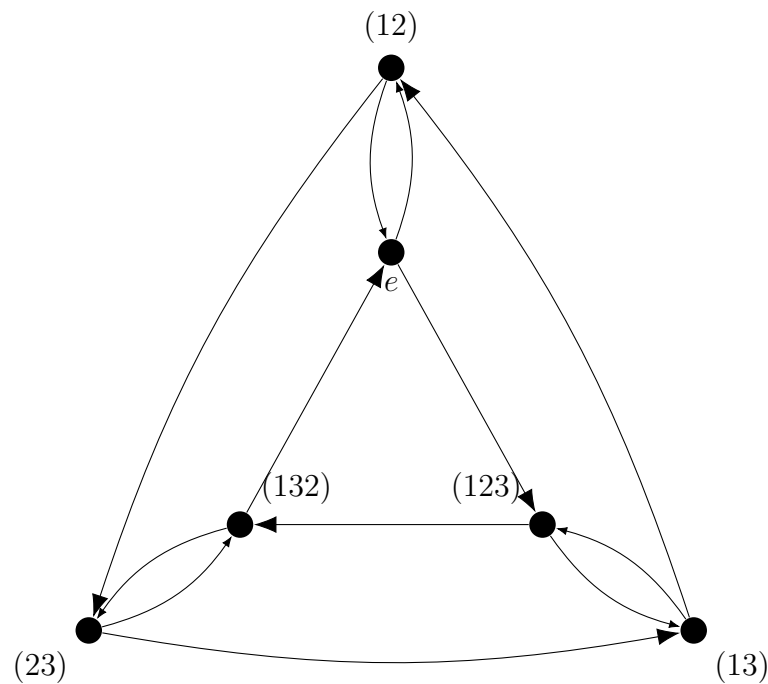


Figure 1.1 - (a) Cayley Graph of S_3 with respect to the generating set $\{(12), (123)\}$.

In Figure 1.1a, the small edges represent the action of the generator (12) , while the long arrows represent the action of the generator (123) . In Figure 1.1b, the blue, red and black arrows are, respectively, associated to the generators (12) , (13) and (23) .

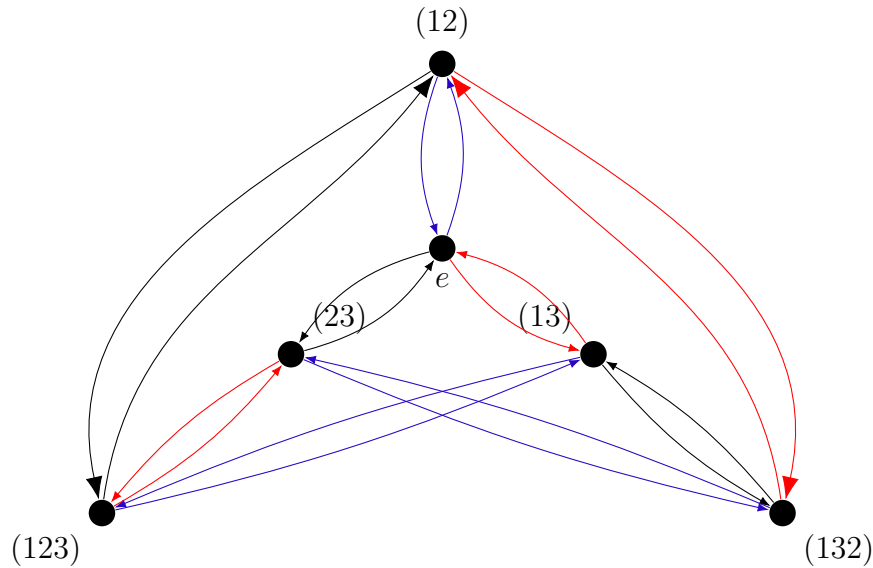


Figure 1.1 - (b) Cayley Graph of S_3 with respect to the generating set $\{(12), (23), (13)\}$.

After this introduction to graphs, we can go on to study free groups, which can be intuitively understood as groups that can be presented without relations and such that any other group is a quotient of one of them. Using symbols, if F is a free group generated by S , then F is presented as $\langle S \mid \rangle$.

Definition 1.2.17. Consider a set M and \bullet a binary operation $\bullet : M \times M \longrightarrow M$. The pair (M, \bullet) is a *monoid* if it is a semigroup with identity, that is, if

1. for all $a, b, c \in M$, associativity $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ holds; and
2. there exists an identity element $e \in M$ such that for every $a \in M$, $a \bullet e = e \bullet a = a$.

Definition 1.2.18. Let S be a set. A finite sequence of (not necessarily distinct) elements of S is a *word* (in S). The set of all words in S (including the empty word), denoted by S^* , together with the operation of concatenation, is the *free monoid* on S .

In the context of studying formal languages, any set S is called an *alphabet*. Being the set of all words in an alphabet the underlying set of a monoid, we are interested in the definition of monoid homomorphisms.

Definition 1.2.19. Consider two monoids M and M' . A *monoid homomorphism*, also called just monoid morphism, is a function $\phi : M \longrightarrow M'$ such that

$$\phi(m_1 m_2) = \phi(m_1) \phi(m_2) \text{ for all } m_1, m_2 \in M$$

and

$$\phi(e_M) = e_{M'}.$$

Clearly, the image of a monoid morphism is a submonoid of its range. When working with monoid morphisms in Chapter 3, we may refer to them as *morphisms* only. Now we go back to the free groups.

Definition 1.2.20. Consider $S = \{x_1, x_2, \dots, x_n\}$, $S \subseteq G$ a group. We denote S^{-1} as the set of the *inverse elements* of S in G .

It is important to notice that we interpret elements of $(S \cup S^{-1})^*$ in two different manners: as elements of a group G (generated by S) and as words in the alphabet $S \cup S^{-1}$, without any operation or inverses. Thus, if $a, b \in S$, we have $baa^{-1} = b$ in G , but $baa^{-1} \neq b$ in $(S \cup S^{-1})^*$.

Notation 1.2.21. We use $baa^{-1} \equiv_G b$ to denote that the equality holds when interpreting b and baa^{-1} as elements of G .

Definition 1.2.22. Consider $S = \{x_1, x_2, \dots, x_n\}$, $S \subseteq G$ a group. A word $w \in (S \cup S^{-1})^*$ is said to be *freely reduced* if it does not have any element $a \in \{S \cup S^{-1}\}$ followed immediately by a^{-1} . This means that w does not have a subword composed only by a generator $x_i \in S$ and its inverse $x_i^{-1} \in S^{-1}$.

Notation 1.2.23. We denote by \bar{S} the set $S \cup S^{-1}$.

Definition 1.2.24. G is a *free group* with basis S if S generates G and there is no nontrivial freely reduced word $w \in \bar{S}^*$ with $w \equiv_G 1$.

Thus, the free group can also be understood as the set of all freely reduced words, with the operation concatenation (followed by reduction if necessary) of words.

We now present a series of results about the free groups whose proofs can be found in any basic literature in the subject of free groups, such as [21] or [25].

Theorem 1.2.25. *There exists F a free group with rank n , for every $n \in \mathbb{N}$.*

Theorem 1.2.26. *Consider a group G , $g_1, g_2, \dots, g_n \in G$ and $X = \{x_1, x_2, \dots, x_n\}$ a basis for F a free group (with rank n). Then there exists a group homomorphism $\phi: F \rightarrow G$ with $\phi(x_i) = g_i$ for every $1 \leq i \leq n$.*

Notice that if $\{g_1, g_2, \dots, g_n\}$ is a set of generators for G , then the map is actually surjective because its image contains all the generators of G . This result can be expressed in the form: for each map $f: X \rightarrow G$, there exists a unique group homomorphism $\phi: F \rightarrow G$ such that $\phi(x_i) = f(x_i)$ for every $x_i \in X$. In other words, ϕ makes the following diagram commute

$$\begin{array}{ccc} F(S) & \xrightarrow{\phi} & G \\ & \searrow f & \nearrow \\ & X & \end{array} .$$

This form of expressing the result is commonly referred to as the universal property of the free groups.

Corollary 1.2.27. *Any two free groups F_1 and F_2 with same rank n are isomorphic.*

This last result allows us to speak about the free group of rank n instead of a free group with rank n . We denote by \mathbb{F}_n the free group with rank n , for $n \in \mathbb{N}$. A free group with infinite rank is denoted by \mathbb{F}_∞ .

Corollary 1.2.28. *Consider a group G generated by S , with $|S| = n$. Then, $G \cong \mathbb{F}_n/H$, where $H = \mathcal{N}(\phi)$ is the kernel of the map ϕ as defined in Theorem 1.2.26, considering as g_i 's the elements of the generating set S .*

We remember that the symbol \cong express the existence of an isomorphism.

Theorem 1.2.29. *The Cayley graph with respect to any generator set of a free group is a tree.*

Theorem 1.2.30 (Nielsen-Schreier). *Every subgroup of a free group is itself a free group.*

Theorem 1.2.31 (Howson). *If G is a free group and H_1, H_2, \dots, H_n is a finite number of finitely generated subgroups of G , then $\bigcap_{i=1}^n H_i$ is a finitely generated free group.*

Definition 1.2.32. Let G be a group and consider a group property \mathcal{P} . If there is a finite index subgroup of G , H , with property \mathcal{P} , then G is said to be *virtually \mathcal{P}* .

We use this last definition mostly for the virtually free groups, but it holds for any other group property, such as being cyclic, abelian, nilpotent or solvable.

1.3 Group Presentations

Theorem 1.2.26 says that if G is a finitely generated group with generators $S = \{g_1, \dots, g_n\}$, then there is a homomorphism ϕ from \mathbb{F}_n , which has basis $X = \{x_1, \dots, x_n\}$, onto G such that $\phi(x_i) = g_i$. Thus, given a word ω in \mathbb{F}_n , it corresponds to some product of generators and their inverses in G . If ω maps to the identity in G , then ω is called a relation in G .

Definition 1.3.1. Consider G a group, $\phi : \mathbb{F}_n \rightarrow G$ a surjective map and $R \subseteq \ker(\phi)$. A subset R of \mathbb{F}_n is said to be a set of *defining relations* for G if the smallest normal subgroup of \mathbb{F}_n that contains R is $\ker(\phi)$.

Note that, if R is a set of defining relations, then every element in the kernel of ϕ can be expressed as a finite product of conjugates of the elements of R and their inverses.

With this in mind, we can define the finite group presentations.

Definition 1.3.2. A group G is said to be *finitely presented* if there exists a finite set of generators S and a finite set of defining relations R for G . In that case, one writes

$$G = \langle S \mid R \rangle = \langle g_1, \dots, g_n \mid \omega_1, \dots, \omega_m \rangle.$$

A group presentation is a choice of generating set and set of relations for a group as above, without the restrictions that S or R be finite, but we will only be interested in groups with finite presentations.

In general, it is not easy to prove that a group has a particular presentation and usually there is no general procedure to prove that a given presentation is related to a given group. Given an arbitrary presentation, it is hard even to determine if the group is infinite, finite or trivial. We do not go very deep into presentation problems, but we will define some groups by means of presentations.

1.4 Languages and Dehn's Problems

We now define formal languages and introduce Dehn's problems for groups.

Definition 1.4.1. Given an alphabet $S = \{x_1, x_2, \dots, x_n\}$, any subset of words in the free monoid $(S \cup S^{-1})^*$ is a *language*.

We previously defined the inverse set S^{-1} of a given alphabet by means of a group generated by S . Nevertheless, the inverse set may be defined without any relation to groups.

Definition 1.4.2. Let $S \neq \emptyset$ be an alphabet and choose a set S^{-1} which has a bijection to S and such that $S \cap S^{-1} = \emptyset$. We denote the image of $s \in S$ under this bijection by s^{-1} .

Formal languages can be studied by means of automata or by means of grammars. We use both of these approaches in this work. The languages can be classified according to many properties they may have. We recall that our main interest lies in the classification of formal languages according to the automata type related to them, that is, the classification given by Chomsky's Hierarchy, first defined by Chomsky in [9].

Definition 1.4.3 (Chomsky's Hierarchy). Chomsky's Hierarchy divides the formal languages in four classes, each of them containing the next one.

- *Type 0*: Recursively countable languages. (Accepted by Turing machines.)

- *Type 1:* Context-sensitive languages. (Accepted by linearly nondeterministic Turing machines.)
- *Type 2:* Context-free languages. (Accepted by nondeterministic pushdown automata.)
- *Type 3:* Regular languages. (Accepted by deterministic finite state automata.)

Context-free and regular languages are our main interest and will be introduced in the following chapters. Let us now define Dehn's Problems for finitely generated groups.

In 1910, Max Dehn introduced some decision problems for groups while studying the fundamental group of closed surface with genus greater than or equal to 2. Decision problems are useful as a first test of how complex a group can be. Given a group G with finite generating set S , Dehn's problems ask if it is possible to algorithmically solve the following questions:

- (a) Given a word w in the alphabet $\bar{S} = \{S \cup S^{-1}\}$, can it be decided if it is the identity element when viewed as an element of G ? That is, is there an algorithm that determines if, given $w \in \{S \cup S^{-1}\}^*$, one has $w \equiv_G e_G$? (*Word Problem*);
- (b) Given $w, w' \in \bar{S}^*$, can it be decided if they represent the same element of G ? That is, can it be decided whether $w \equiv_G w'$? (*Equality Problem*);
- (c) Given two words w_1, w_2 in the alphabet \bar{S} , can it be decided if they are conjugate as elements of G ? That is, if w_1 and w_2 represent group elements g and h , respectively, can it be decided if there is a $z \in G$ such that $g = zhz^{-1}$? (*Conjugacy Problem*);
- (d) If G' is another group, with generators S' and relations R' , can it be decided if the groups G and G' are isomorphic? (*Isomorphism Problem*).

The computability theory involved in the definition of what the existence of an algorithm means is extensive and we do not focus on it. We consider the existence of an algorithm for resolving a given decision problem A to mean the existence of a Turing machine (which are a specific type of automata we do not explore) connected with a black box (usually called an *oracle*) such that the system "machine + oracle" is able to resolve the decision problem. The oracle is supposed to be able to give a solution, not necessarily a computational one, for some related decision problem. Thus, the desired algorithm exists if there is a Turing machine that can reduce the main problem A to a problem B which is solvable by some oracle.

It is possible to show that the Word Problem and the Equality Problem are equivalent. The following results can be found in Chapter 5 of [21].

Proposition 1.4.4. *Given a group G with finite generating set S , Dehn's word problem is solvable for G if and only if the equality problem is solvable for G .*

However, these problems are not always decidable.

Theorem 1.4.5. *There are finitely presented groups G such that there is no algorithm determining if a given word in \overline{S} represents (in the group G):*

1. *the identity;*
2. *an element in the center of G , $Z(G)$;*
3. *an element in the commutator subgroup;*
4. *a finite order element.*

It is possible to relate a property of a Cayley graph of a group and its word problem.

Definition 1.4.6. Let Γ be a connected graph and $v, w \in V(\Gamma)$. The *length of a path* between v and w is equal to the number of edges in such path. The *distance* between v and w is equal to the length of the path of minimal length from v to w . We denote the distance by $d(v, w)$.

Definition 1.4.7. Consider a connected graph Γ and $v \in V(\Gamma)$. The *ball* centered in v with radius n , $B(v, n)$, is the subgraph containing all paths beginning in v with length lower or equal to n .

Definition 1.4.8. A Cayley graph $\Gamma_{G,S} = (V, E)$ is said to be *constructible* if there exists a total order in its set of vertices, \leq , so that given $v \in V \setminus \{e_G\}$, then there exists $y \in V$ such that

1. $y < v$,
2. y is adjacent to v ; and
3. y is adjacent to w for any $w \in V$ which is adjacent to v such that $w < v$.

This means that, given $B(v, n)$ for some $n \in \mathbb{N}$, it is possible to construct $B(v, n + 1)$ by adding vertices one by one in a finite amount of steps.

Proposition 1.4.9. *A group G with finite generating set S has solvable Word Problem if and only if it is possible to decide which words in \overline{S} correspond to a closed path in the Cayley graph $\Gamma_{G,S}$.*

We finish this chapter with the following result which shows the connection between the Word Problem and group theoretic properties.

Theorem 1.4.10. *A group G with finite generating set S has solvable Word Problem if, and only if, the Cayley graph $\Gamma_{G,S}$ is constructible.*

Further study of these problems is not within our goals, but the interested reader may consult [21]. However, the Word Problem induces the definition of a language in the alphabet S which is of extreme importance to our work. Later in our work we present some interesting results concerning these problems.

2. Anisimov's Theorem

Here we present the definition of a finite state automaton. After that, we see some basic properties concerning both finite state automata and the languages that they accept, that is, regular languages. Finally, we present the proof of Anisimov's result. All the work in this Chapter was done following [21].

2.1 Finite State Automata

Intuitively, an automaton is a directed graph \mathcal{M} with an associated alphabet X and such that:

1. There exists a subset of $V(\mathcal{M})$ whose elements are called initial (or start) states.
2. There exists a subset of $V(\mathcal{M})$ whose elements are called accepted states.
3. Directed edges are labelled by elements of X .

The name automaton comes from the fact that such a graph is constructed to simulate the behavior of a computer. Intuitively, a finite state automaton is a finite directed graph with decorated edges. Formally, however, these automata are an abstract 5-uple.

Definition 2.1.1. A *nondeterministic finite state automaton* is a quintuple $\mathcal{M} = (Q, X, \delta, I, F)$, where:

1. Q is a finite set, called the set of states of \mathcal{M} ;
2. X is a finite set, called the alphabet of \mathcal{M} ;
3. $\delta \subseteq (Q \times X) \times Q$ is a finite set of transition relations;
4. $I \subseteq Q$ is called the set of initial (or start) states of \mathcal{M} ;
5. $F \subseteq Q$ and is called the set of final (or accepted) states of \mathcal{M} .

Although the transition relations are defined as 3-uples, they can be written as a multivalued function $\delta: (Q \times X) \longrightarrow Q$. Transition relations follow the formation of words accepted by the automaton. Let us picture a nondeterministic finite state automaton as a graph: the vertices are the states; and for each $(u, s, v) \in \delta$, there is a directed edge leaving the vertex u labeled by s and arriving at the vertex v . See Example 2.1.6.

Definition 2.1.2. The *language accepted* by an automaton \mathcal{M} is formed by all words $w \in X^*$ corresponding to directed paths in \mathcal{M} , beginning in an initial state and ending in an accepted state. We denote it by $\mathcal{L}(\mathcal{M})$.

A finite state automaton can be deterministic or nondeterministic. The determinism appears when the automaton has exactly one initial state and, picturing it as a graph, there are no two edges beginning in the same vertex and having the same label.

Definition 2.1.3. A *deterministic finite state automaton* (DFA) is a quintuple $\mathcal{M} = (Q, X, \delta, q_0, F)$, where:

1. Q is a finite set, called the set of states of \mathcal{M} ;
2. X is a finite set, called the alphabet of \mathcal{M} ;
3. $\delta \subseteq (Q \times X) \times Q$ is a finite set of transition relations such that
 - for each $q \in Q$ and $\omega \in X$, there is at most one transition of the form $((q, \omega), *)$, $* \in Q$;
4. $q_0 \in Q$ is called the initial (or start) state of \mathcal{M} ;
5. $F \subseteq Q$ and is called the set of final (or accepted) states of \mathcal{M} .

The name of the DFAs come from the fact that, given a word accepted by the automaton, there is a unique path that must be followed in the automaton associated graph in order to form that word.

Definition 2.1.4. The language accepted by a DFA is called a *regular language*.

Remark 2.1.5. For a language to be regular, it must be exactly the language accepted by a DFA and not a subset of the language accepted by it.

Example 2.1.6. We briefly remind that the most common representation of a graph is made by using points and lines. Thus, a directed edge (an edge that has an initial and a final vertex) is an arrow, going from one point to the other. Labeling an edge is nothing more than putting a symbol above or below the arrow.

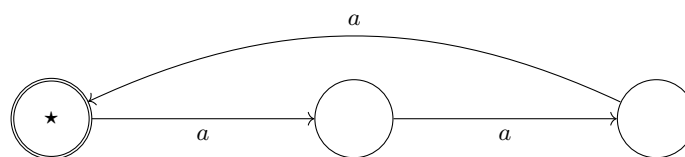


Figure 2.1 – A deterministic automaton

The graph in Figure 2.1 represents a deterministic automaton (the only start state is denoted by \star), which has only one accepted state (denoted by two circles). Note that the accepted state can be the same as the start state (and it is in this example). The alphabet considered could be any set S that contains a .

Consequently, the language accepted by the automaton presented here is $\{a^{3n} \mid n \in \mathbb{N}\}$. And it is a regular language, since the automaton is deterministic.

Though it is not among our goals to prove it, there is an interesting result in [21], Chapter 7, concerning the deterministic and nondeterministic automata:

Theorem 2.1.7. *The set of languages accepted by nondeterministic finite state automata is the same as the set of regular languages.*

Using the above result, some closure properties for the regular languages can be proven.

Theorem 2.1.8. *Let K and L be two regular languages on the same alphabet S . Then, the following are regular languages:*

1. *The complementary language $S^* \setminus K$.*
2. *The union, $K \cup L$.*
3. *The intersection, $K \cap L$.*
4. *The concatenation, $KL = \{\omega_K \omega_L \mid \omega_K \in K \text{ and } \omega_L \in L\}$.*
5. *The infinite union $K \cup KK \cup KKK \cup \dots$.*

Before proving this result, we introduce a new definition.

Definition 2.1.9. We say that a finite state automaton with associated alphabet S is *complete* if, for every vertex in the automaton, $v \in V$, and for every $a \in S$, there exists an edge leaving v labelled by a .

We note that, in the above definition, there are no restrictions on the tails of the edges, they may be loops or not.

Proof of Theorem 2.1.8. We first note that, given a DFA \mathcal{M}' with associated alphabet S , we can construct a complete DFA that accepts the same language as \mathcal{M}' . In order to do that, we add, for each vertex $v \in V'$, as many edges as needed in order to have the property that for all $v \in V, a \in S$, there exists an edge leaving v labelled by a . The new added edges all leave v and go to a new vertex $q \notin V'$, for each $v \in V'$. The vertex q have one loop edge for each $a \in S$, labelled by that same a , and is not an accepted state. Then,

we have a complete DFA \mathcal{M} with set of vertices $V = V' \cup \{q\}$ that accepts the same language as \mathcal{M}' .

(1) Assume K is accepted by a DFA \mathcal{M}' . We construct from \mathcal{M}' the complete automaton \mathcal{M} , which accepts the same language K . Consider the automaton \mathcal{M}^c formed by taking the complement of the set of accepted states of \mathcal{M} as accepted states, but with same set of edges and vertices. Since \mathcal{M} is finite, so is \mathcal{M}^c . The sets of accepted states and initial states of \mathcal{M}^c are both well-defined, since they were for \mathcal{M} . Thus, \mathcal{M}^c is a DFA. We need only show that it accepts the language $S^* \setminus K$.

First, we notice that the associated alphabet of \mathcal{M}^c is still S , so that $\mathcal{L}(\mathcal{M}^c) \subseteq S^*$. Secondly, given that \mathcal{M} is deterministic, for every word $\omega \in \mathcal{L}(\mathcal{M})$, there is only one path in \mathcal{M} beginning in the start state and ending in an accepted state which form ω . In \mathcal{M}^c , this same path still form ω , but it ends in a non-accepted state. Since there is no change in the set of initial states, then there is no other way of forming ω in \mathcal{M}^c , otherwise there would have to be a path in \mathcal{M} labelled by ω but ending in a non-accepted state, which is impossible because of the deterministic aspect of \mathcal{M} . Thus, $\omega \notin \mathcal{L}(\mathcal{M}^c)$ so that $\mathcal{L}(\mathcal{M}) \cap \mathcal{L}(\mathcal{M}^c) = \emptyset$, which means that $\mathcal{L}(\mathcal{M}^c) \subseteq S^* \setminus K$.

Finally, take $\eta \in S^* \setminus K$. We have made no changes in edges when going from \mathcal{M} to \mathcal{M}^c , so all the paths are exactly the same in both graphs. Now, because \mathcal{M} is complete but does not accept η , there must be a path forming η in \mathcal{M} which ends in a non-accepted state. The deterministic aspect of \mathcal{M} guarantees that such path is unique. Thus, when changing all non-accepted states into accepted states, we change η into an accepted word. Then, $\eta \in \mathcal{L}(\mathcal{M}^c)$ and we have that \mathcal{M}^c accepts the complementary language.

(2) Consider the DFAs \mathcal{M}_L and \mathcal{M}_K , which accepted the languages L and K , respectively. Let us unite \mathcal{M}_L with \mathcal{M}_K in a disjoint manner (by simply considering both of them, together, as one single automaton). The union is a nondeterministic finite state automaton, because there are two start states. Then, we have a nondeterministic finite state automaton that accepts the language $K \cup L$. The previous theorem guarantee that this means $K \cup L$ is a regular language.

(3) Write $K \cap L = S^* \setminus \{(S^* \setminus K) \cup (S^* \setminus L)\}$. The result then follows from (1) and (2).

(4) Consider the DFAs \mathcal{M}_L and \mathcal{M}_K , which accept the languages L and K , respectively. Let us unite \mathcal{M}_L with \mathcal{M}_K in a disjoint manner (by simply considering both of them, together, as one single automaton). Now we declare that the only start state of this new automaton \mathcal{M} is the start state of \mathcal{M}_K . We then add an edge from each accepted state in \mathcal{M}_K to the vertex that used to be the start state of \mathcal{M}_L , labelled by a symbol ϵ outside our alphabet S . This transforms \mathcal{M} into a finite state automaton, which could be nondeterministic or not. Going through the new edge adds nothing to the word being read by a path in the automaton, so that the language accepted by \mathcal{M} is exactly KL .

Since \mathcal{M} is a finite state automaton, Theorem 2.1.7 finishes our proof regardless of the determinism of \mathcal{M} .

(5) Consider an automaton \mathcal{M} constructed by starting with \mathcal{M}_K , the DFA accepting K , and adding edges from each accepted state to the initial state of \mathcal{M}_K . We label these edges by ϵ . As in the previous item, these edges add nothing to the words being read, but they give a path from the end of a word to the beginning of another, thus the language accepted is exactly $K \cup KK \cup KKK \cup \dots$. Again, since \mathcal{M} is a nondeterministic finite state automaton, the previous theorem finish our proof. \square

2.2 Main Theorem

As noted in Chapter 1, the Word Problem for groups induces the definition of a language in the alphabet S which is fundamental for this work. Let us now define that language.

Definition 2.2.1. Consider a group G generated by a finite set S . The set $\text{WP}(G, S) = \{w \in \{S \cup S^{-1}\}^* \mid w \equiv_G e_G\}$ is defined as the *language of the word problem* of G with respect to S . We often omit the letter S (and G) when it is clear which set of generators (and group) is being considered.

The above definition makes solving Dehn's word problem the same as discovering if $w \in \text{WP}(G, S)$. The importance of this language for us lies in the classification of a class of finitely generated groups.

Theorem 2.2.2 (Anisimov). *Consider a group G with finite generating set S . Then $\text{WP}(G, S)$ is a regular language if, and only if, G is finite.*

Proof. Let us show first how to build a DFA accepting $\text{WP}(G, S)$ and then show the other direction.

\Leftarrow If G is finite, then the Cayley graph with respect to $\{S \cup S^{-1}\}$ is finite. We now regard the Cayley graph as a DFA, by declaring the identity element as the unique initial state as well as the unique accepted state. The language accepted by this DFA is $\text{WP}(G, S)$.

\Rightarrow Conversely, suppose that $\text{WP}(G, S)$ is the language generated by a DFA \mathcal{M} . Without loss of generality, assume that every vertex in \mathcal{M} is connected to an accepted state by a directed path (otherwise, this vertex is not part of any accepted word and can be removed from the automaton).

Let w and w' be two words formed in the graph of \mathcal{M} that have paths ending in the same vertex $z \in \mathcal{M}$, which may or may not be an accepted state. By our previous

assumption, there exists a path from z to an accepted state. Consider the word v formed by such a path. Then both wv and $w'v$ are accepted by \mathcal{M} . By hypothesis, this means that $wv \equiv e_G \equiv w'v$ as elements in G . Then, as G is a group, one can apply right cancellation and obtain that $w \equiv w'$ in G .

We have obtained that, given two words described by paths ending at the same vertex in the graph of \mathcal{M} , then they represent the same element of G . This implies that G has order less or equal to the number of states in the graph of \mathcal{M} . But \mathcal{M} is a DFA, meaning that its graph has a finite number of states. Thus, G has finite order. \square

3. Muller and Schupp's Theorem

3.1 Pushdown Automata

3.1.1 Context-Free Languages

We now introduce the concept of pushdown automata. As we have briefly mentioned, they are used to define the class of context-free languages, which is used to classify another class of groups, that of virtually free groups. In this section, some closure properties are presented, as well as a particular case leading to a less general, but interesting result (Herbst's Theorem) in the Hierarchy we are constructing for groups.

Definitions and properties can be found in more detail in [14] or, for more advanced readers, in [3, 11]. Herbst's intermediate result can be found in [12].

Informally speaking, a pushdown automaton is a nondeterministic finite state automaton \mathcal{M} with an associated alphabet S and such that:

- There exists a vertex $v \in V(\mathcal{M})$ which is called initial state.
- There exists a subset of $V(\mathcal{M})$ whose elements are called accepted states.
- Directed edges are labelled by elements of S .
- It has two other related alphabets, the stack and the tape alphabet.
- It has a set of transition relations.

These automata cannot be pictured only as a graph, as was the case for the finite state ones. The pushdown automata can be interpreted as a machine composed by three pieces: a sequence of symbols, called the input tape; a finite graph dictating what the machine does; and another sequence of symbols, called the stack.

Definition 3.1.1. A *nondeterministic pushdown automaton* is a 7-tuple

$$\mathcal{M} = (Q, X, \Sigma, \delta, q_0, \$, F),$$

where

1. Q is a finite set, called the set of states of \mathcal{M} ;
2. X is a finite set, called the tape alphabet of \mathcal{M} ;

3. Σ is a finite set, called the stack alphabet of \mathcal{M} ;
4. $\delta \subseteq ((Q \cup \{\varepsilon\}) \times (X \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})) \times (Q \times \Sigma^*)$ is a finite set of transition relations;
5. $q_0 \in Q$ is called the initial (or start) state;
6. $\$ \in \Sigma$ is called the bottom of the stack;
7. $F \subseteq Q$ is the set of accepted states.

As in the case of the finite state automata, the transition relations defined as 5-uples can be written as if their set was a function, in order to improve the notation:

$$\delta: ((Q \cup \{\varepsilon\}) \times (X \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})) \longrightarrow (Q \times \Sigma^*).$$

Notice, however, that δ is not truly well defined as a function, because of the nondeterminism.

The main difference between pushdown and finite state automata is given by the existence of the stack, which works as a memory for the automaton. The stack is a storage device represented by a one-sided pile with countably many empty spaces, as is the input tape. The input tape is filled with a finite word over the tape alphabet X , which will give instructions to the automaton, as it reads each letter of the word, about which transition to follow. The stack begins with a given word or symbol at the top of the pile and has two possible operations, which it does accordingly to the instructions on the transitions. The possible operations are:

1. to replace symbols of the stack, reading its top letter, deleting it at the same time and afterwards adding the word (possibly empty) given by the last followed transition;
and
2. to fill the stack, not deleting any of its symbols and adding the word (possibly empty) given by the last followed transition.

We call the current symbol being read in the input tape the *input letter*. Thus, pushdown automata have to consider the current state, the input letter and the top (last symbol) of the stack before making a transition, while the finite state automata consider only the current state.

It is natural to define, as was done with the DFA, the class of languages which are accepted by pushdown automata:

Definition 3.1.2. A language $\mathcal{L} \subseteq X^*$ is said to be *accepted* by a pushdown automaton $\mathcal{M} = (Q, X, \Sigma, \delta, q_0, \$, F)$, if \mathcal{M} reaches an accepted state when it finishes reading ω from the input tape, for any $\omega \in \mathcal{L}$. The languages accepted by pushdown automata are called *context-free languages*.

The easiest way to completely understand this concept is by means of an example.

Example 3.1.3. We usually represent an automaton as a graph whose vertices are the states. We denote the start state by an arrow coming into it from nowhere and the accepted states by two circles. The transition relations are represented by the directed and labeled edges, being the labels of the form $a, b \rightarrow c$, with $a \in X \cup \{\epsilon\}$ and $b, c \in \Sigma \cup \{\epsilon\}$. We call the entries a, b and c by first, second and third positions of the relation (or of the label), respectively.

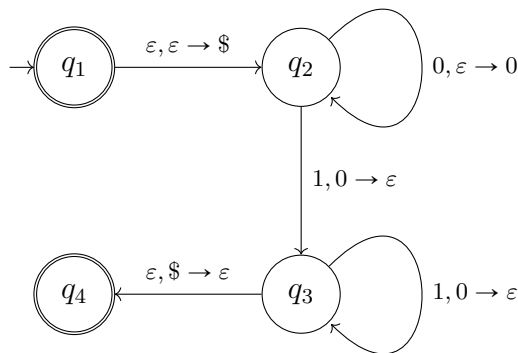


Figure 3.1 – A pushdown automaton.

The input tape consists of a sequence of letters on the tape alphabet X , which is given to the automaton to read. The automaton reads the sequence and, together with the transition relations, makes changes to the stack pile. The stack is a sequence of symbols in Σ . We interpret this sequence as a pile of symbols and we call the last element in the sequence the top of the stack (pile). The end of the pile is the first symbol in the sequence. The automaton in Figure 3.1 represents a pushdown automaton

$$\mathcal{M} = \{\{q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0\}, \delta, q_1, \$, \{q_1, q_4\}\},$$

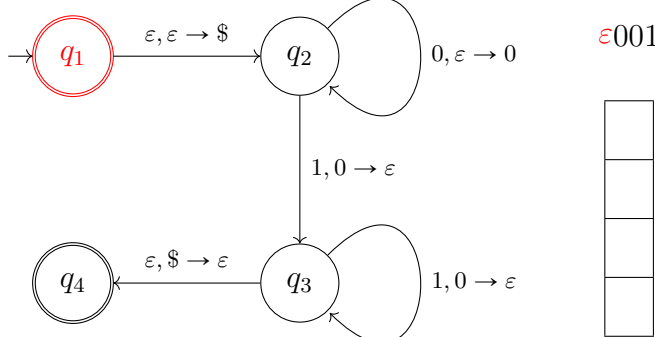
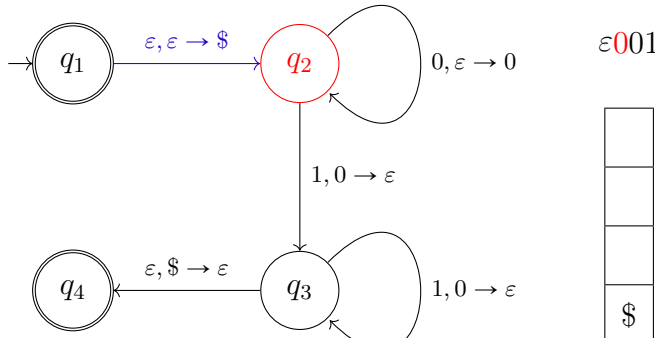
which accepts the language $\{0^n 1^n \mid n \geq 0\}$. To understand that, it is necessary to comprehend the workings of the transition relations δ when pictured in the graph.

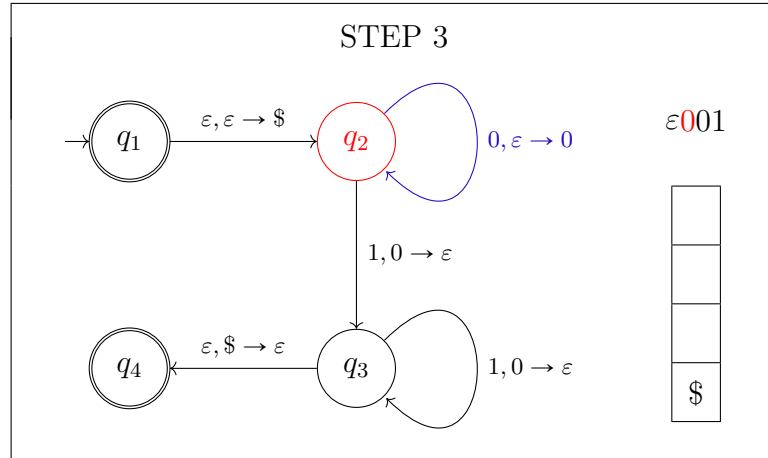
Writing " $a, b \rightarrow c$ " means that, if on the input tape there is an a and on top of the stack pile there is a b , then the automaton can follow that edge and it replaces b by c on

the stack. After that, the automaton goes to the next symbol on the input tape and obey the transitions that begin in the state where it has arrived with the last change. When there is an ε in the transition relation, it is understood that there is no symbol: If there is an ε in the first position, the machine may make the transition without reading anything from the input tape; in the second position, the machine does not need to read or replace any symbol from the stack pile; in the third position, the automaton processes the element it reads and erase it, but does not write any symbol on the stack. Thus, the mechanism can eventually reach the $\$$ symbol, which marks the end of the stack pile.

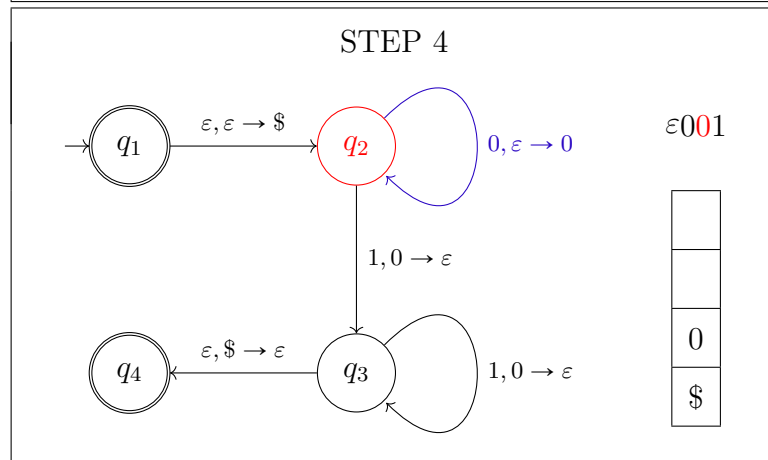
For a clearer explanation, let us give the automaton on Figure 3.1 the word 001 written in the input tape and describe its behavior, see Table 1.

Table 1 – Computation Example with Input Word 001

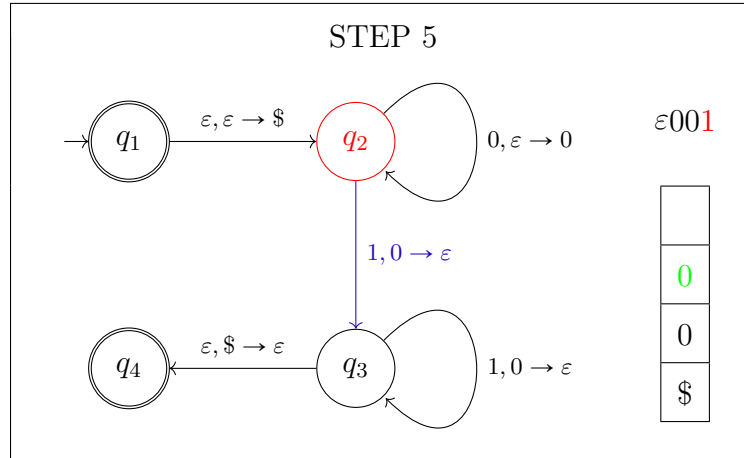
<p style="text-align: center;">STEP 1</p> 
<p>The automaton \mathcal{M} begins at state q_1. Leaving q_1 there is only one edge, with ε in the first position of the label, which is the same as requiring that no symbol is read from the input tape. Thus, this condition is automatically satisfied. \mathcal{M} has yet to verify if the second condition is satisfied, by reading the symbol in the second position of the edge label. The second position has an ε, which means "independent of the stack's contents". This condition is trivially satisfied, therefore \mathcal{M} can indeed go through this edge and can apply the correspondent changes to the stack.</p>
<p style="text-align: center;">STEP 2</p> 
<p>\mathcal{M} followed the blue edge and arrived at state q_2. First, crossing the edge means applying some change to the stack: the third entry of the transition must be added to the stack, in this case a $\\$. Now \mathcal{M} reads the next symbol on the input tape, which is a "0". Thus the automaton has to look for some edge with a "0" in the first position of the label.</p>



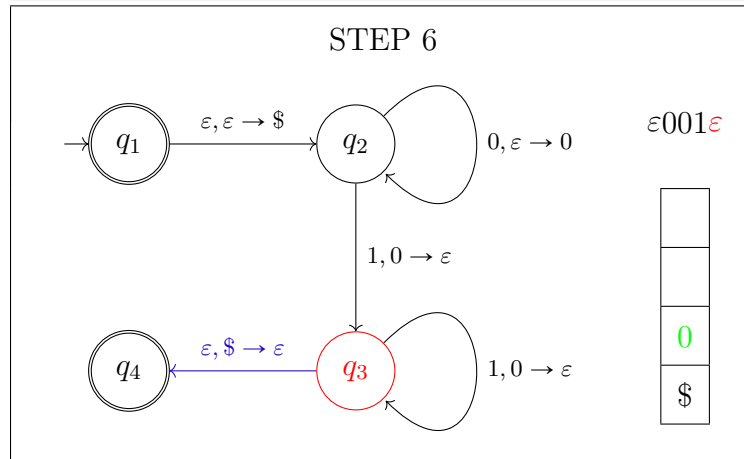
The only edge with 0 as first symbol in the label is the now blue edge. The second symbol from the label is an ϵ and therefore \mathcal{M} can follow this edge without worrying about the contents of the stack, again because the ϵ in the second position means "independence of the stack's contents". The third symbol in the label, 0, tells \mathcal{M} that, once it follows this edge, a "0" has to be added to the stack.



\mathcal{M} added a 0 to the stack, because the last followed edge required that. Next, it goes on to the next symbol in the input tape, which is again 0. Since the blue edge, which \mathcal{M} followed in last step, leads it back to the state q_2 , the only possible edge satisfying the first transition condition (that is, having first symbol equal to the current input symbol being read) is again the blue edge. The second condition for following that edge is trivial and therefore \mathcal{M} follows it.



Again, \mathcal{M} adds a 0 to the stack in the last step. The next input symbol is now a 1, which means the one possible choice of edge is the blue one. To follow that edge, \mathcal{M} has to read a 0 on the top of the stack, because the edge label has a 0 in the second position. But that is actually the case, as stressed in green. Hence, \mathcal{M} follows the blue edge and has to erase the 0 it was reading on the top of the stack. Why? Because the third entry in the label of the blue edge is an ϵ , which in the third position means "erase what was read from the stack".



\mathcal{M} is now at the state q_3 and is reading ϵ from the input tape, because the input word has ended. This means that the new blue edge is the one possible choice. But, in order to follow the blue edge, \mathcal{M} has to be both reading ϵ from the input and $\$$ on the stack, which is not the case. As stressed in green, the stack has a 0 on its reading position (the top). Thus, the edge cannot be followed and the automaton has finished reading the input word at state q_3 , which is not an accepted state. Therefore, the word 001 is not accepted by \mathcal{M} .

If we gave as input word the word 0011 instead of 001, then there would be another step between steps 5 and 6 in Table 1, where the loop edge from q_3 to itself would be followed and the second 0 would be erased from the stack. In that case, the blue edge of step 6 could be followed and the automaton would finish reading the input in an accepted state. Therefore, 0011 is accepted by \mathcal{M} . Finally, notice that the behavior of the automaton will reject any word where there are 0s after a 1, because once the first 1 is read, the automaton reaches state q_3 and cannot read any other 0. From that and the computation example, it becomes clear that the accepted language is indeed $\{0^n 1^n \mid n \geq 0\}$.

It is also natural to ask about the deterministic case. Deterministic pushdown automata are defined below.

Definition 3.1.4. A *deterministic pushdown automaton* is a 7-tuple

$$\mathcal{M} = (Q, X, \Sigma, \delta, q_0, \$, F),$$

where

1. Q is a finite set, called the set of states of \mathcal{M} ;
2. X is a finite set, called the tape alphabet of \mathcal{M} ;
3. Σ is a finite set, called the stack alphabet of \mathcal{M} ;
4. $\delta \subseteq ((Q \cup \{\varepsilon\}) \times (X \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})) \times (Q \times \Sigma^*)$ is a finite set of transition relations such that
 - for every $q \in Q$, $x \in X$ and $a \in \Sigma$, at most one of the transitions $((q, x, a), *)$, $((q, x, \varepsilon), *)$, $((q, \varepsilon, a), *)$, $((q, \varepsilon, \varepsilon), *)$ exists, with $*$ $\in (Q \times \Sigma^*)$;
5. $q_0 \in Q$ is called the initial (or start) state;
6. $\$ \in \Sigma$ is called the bottom of the stack;
7. $F \subseteq Q$ is the set of accepted states.

The language accepted by a deterministic pushdown automaton is called a *deterministic context-free language*.

Remark 3.1.5. The formal definition of transitions uses heavy notation. In order to make it easier to visualize what the transitions are doing, we frequently use the notation $\delta(q, x, \sigma) = (\tilde{q}, \tilde{\sigma})$ instead of $((q, x, \sigma), (\tilde{q}, \tilde{\sigma}))$. Notice that contrary to the nondeterministic case, in the deterministic case δ can be understood as a well defined function. When represented inside automata, the transitions have yet another notation, as seen in the automaton of Figure 3.1.

Remember that in the Chomsky's Hierarchy, the regular languages were presented as a subset of the context-free languages. Indeed, for any given regular language, consider its generating finite state automaton with associated alphabet X . For each edge in that automaton, take the label $x \in X$ and change it with $x, \varepsilon \mapsto \varepsilon$, obtaining a new automaton, where the set of states, the accepted states and the initial states are the same. This new automaton is a pushdown automaton accepting the original regular language, which is therefore context-free. Notice that the stack alphabet in such pushdown automaton could be any set. But the set of context-free languages is strictly larger than the set of regular languages. Take as example the language of Example 3.1.3, $\mathcal{L} = \{0^n 1^n \mid n \in \mathbb{N}\}$, which is context-free, and observe that \mathcal{L} is not a regular language. This can be easily proved using the so-called Pumping Lemma for Regular languages, which the interested reader can find in any formal languages theory book, such as [28] or [21]. We do not state the lemma or the proof because they are extensive and the involved techniques are not necessary for the main goal of this work.

In order to prove some results presented in Section 3.2, we need another definition of context-free languages. For that, we construct context-free grammars.

Definition 3.1.6. A *context-free grammar* $\mathcal{G} = (V, X, P)$ consists of:

- an alphabet X of *terminal elements*;
- an alphabet V of *non-terminal elements*, where V is disjoint of X ;
- a finite set of *productions*, $P \subset V \times (V \cup X)^*$.

A production $(\alpha, \beta) \in P$ is written in the form $\alpha \rightarrow \beta$. Productions with same first entry are put together as follows:

$$\left. \begin{array}{l} \alpha \rightarrow \beta_1, \\ \alpha \rightarrow \beta_2, \\ \vdots \\ \alpha \rightarrow \beta_n \end{array} \right\} \alpha \rightarrow \{\beta_1, \beta_2, \dots, \beta_n\}.$$

Now we have to know which languages are generated by a grammar.

Definition 3.1.7. Given $\mathcal{G} = (V, X, P)$ a context-free grammar and $f, g \in (V \cup X)^*$, we write

$$f \xrightarrow[\mathcal{G}]{*} g$$

if and only if there exist $p \geq 0$, $p \in \mathbb{N}$, and $f_0, f_1, \dots, f_p \in (V \cup X)^*$, such that $f = f_0, g = f_p$ and there are factorizations

$$f_{i-1} = u_i \alpha_i v_i, f_i = u_i \beta_i v_i$$

where $\alpha_i \in V$, $u_i, \beta_i, v_i \in (V \cup X)^*$ and $\alpha_i \rightarrow \beta_i \in P$, for all $i = 1, 2, \dots, p$.

A language in X^* generated in \mathcal{G} by $f \in (V \cup X)^*$, is of the form

$$L_{\mathcal{G}}(f) = \{\omega \in X^* \mid f \xrightarrow{\mathcal{G}}^* \omega\}.$$

If a language in X^* is generated in \mathcal{G} by some $f \in (V \cup X)^*$, then it is called a *context-free language*.

The division of non-terminal and terminal elements, as well as the idea of productions, is related to the fact that we want our grammar to construct a particular set of words. The non-terminal elements behave like common states of the pushdown automata: one can travel through them, but cannot end the reading of a word in one of them. The terminal elements are the ones allowed to remain in the final word generated by the grammar, so they behave like the accepted states. The productions play the part of transitions: they are responsible for taking non-terminal elements into words composed only by terminal elements. Although context-free grammars do not work as automata do, there is a certain similarity in the intuitive idea of their behavior. The main difference we encourage the reader to keep in mind is that the elements of a grammar, unlike the states of an automaton, are already seen like letters, which are changed into other letters, according to the productions, until arriving at a string of terminal (accepted) letters.

It is common that we deal with a grammar with the intention of studying only one of its possible generated languages. In this case, it is usual to denote by $S \in (V \cup X)^*$ the element which generates the wanted language and say that the grammar generates only the language of words obtained beginning at S , i.e., the grammar generates only the language of interest. The element S is then called the *start symbol* of the grammar.

Definitions 3.1.2 and 3.1.7 are equivalent, as shown in Chapter 2, Section 2 of [28]. We do not present the proof, as it is too technical for our purposes, but we present an example and a general construction to obtain a pushdown automaton from a context-free grammar. We use both definitions in this work, choosing the more convenient one for each result.

Example 3.1.8. Let $X = \{(,)\}$ be the alphabet consisting of the symbols "(" and ")". The language of all properly matched parentheses, known as the *Dyck Language*, is formally defined as the language

$$\mathcal{L} = \{\omega \in X^* \mid \text{the number of } (\text{ in } \omega \text{ is the same as the number of }) \text{ in } \omega; \text{ and any prefix } u \text{ of } \omega \text{ contains at most as many }) \text{ as there are } (\text{ in } u \}.$$

A generating context-free grammar for \mathcal{L} is $\mathcal{G} = (\{S\}, \{(,)\}, P)$, where P is the set of productions:

$$S \rightarrow \{SS, (S), \varepsilon\},$$

with ε representing the empty string. The equivalent generating pushdown automaton is presented in Figure 3.2.

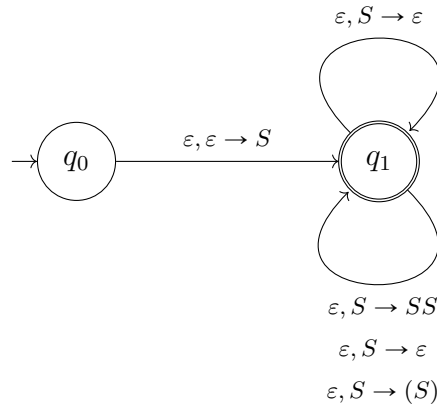


Figure 3.2 – Pushdown automaton accepting the Dyck language.

The construction of a pushdown automaton equivalent to a given context-free grammar can always be done following an algorithm: Given any context-free grammar $\mathcal{G} = (V, X, P)$, construct a pushdown automaton with two states, q_0 (the initial state) and q_1 (the accepting state), and with transition relations

- (i) $(q_0, \varepsilon, \varepsilon, q_1, S)$, where $S \in V$ is a special non-terminal symbol called the start symbol of the grammar;
- (ii) $(q_1, \varepsilon, A, q_1, \alpha)$, for each production $A \rightarrow \alpha$ in \mathcal{G} (called an *expand* transition);
- (iii) $(q_1, a, a, q_1, \varepsilon)$, for each terminal symbol a in \mathcal{G} (called a *match* transition);

The stack alphabet is $X \cup V$, while the input alphabet is X . The first transition guarantees that the stack begins with the symbol S (which acts as start symbol for the grammar) written in it. The information about the start symbol of a context-free grammar is always given together with the grammar, though it is conventional to assume that it is the symbol S . Notice that the constructed pushdown automaton is not deterministic.

For the next results, let $\mathcal{G} = (V, X, P)$ be a context-free grammar. The proofs are either too simple or too technical to be of interest for this work, but can be found in [3], chapter 2, or [28], chapter 2. If L and M are two languages, then LM denotes the concatenation language, as defined in 2.1.8.

Proposition 3.1.9. *For any $f_1, f_2 \in (V \cup X)^*$, $L_{\mathcal{G}}(f_1 f_2) = L_{\mathcal{G}}(f_1) L_{\mathcal{G}}(f_2)$.*

Proposition 3.1.10. *Let $\beta \in V$. Then*

$$L_{\mathcal{G}}(\beta) = L_{\mathcal{G}}(\{\alpha \in V \mid \beta \xrightarrow[\mathcal{G}]{*} \alpha\}).$$

For introducing the main theorem about closure properties of context-free languages, we first need a few definitions.

Definition 3.1.11. Let A, B be two alphabets and $\alpha : A^* \rightarrow B^*$ be a monoid morphism. Then α is:

- *alphabetic*, if $\alpha(A) \subset B \cup \{1\}$;
- *strictly alphabetic*, if $\alpha(A) \subset B$.

Definition 3.1.12. Let $\alpha : A^* \rightarrow \mathcal{P}(B^*)$ be a monoid morphism, where $\mathcal{P}(B^*)$ denotes the power set of B^* , which is a monoid under the intersection of subsets. Then α is called a *substitution* (from A^* to B^*) and verifies:

- $\alpha(a) \subseteq B^*$, for any $a \in A^*$;
- $\alpha(1) = \{1\}$;
- $\alpha(uv) = \alpha(u)\alpha(v)$, for any $u, v \in A^*$.

If, furthermore, α satisfies

- $\alpha(a)$ is a context-free language, for any $a \in A^*$,

then α is a *context-free substitution*.

Theorem 3.1.13. *Let L and M be two context-free languages, with alphabets X and \tilde{X} , respectively. Then the following are context-free languages:*

1. $L \cup M$;
2. $\Psi(L)$, for any $\Psi : X^* \rightarrow Y^*$ a morphism, being Y any alphabet;
3. L^* ;
4. LM ;
5. $L \cap M$, if M is a regular language and $\tilde{X} = X$;
6. $\theta(L)$, being θ a context-free substitution;
7. $\Psi^{-1}(L)$, for any $\Psi : Y^* \rightarrow X^*$ an alphabetic morphism.

We prove, due to length limitations, only the more interesting closure properties. The other proofs can be found in Chapter 2 of [3].

Proof. (2). Consider $\mathcal{G} = (V, X, P)$ the context-free grammar such that $L = L_{\mathcal{G}}(v)$ for some $v \in (V \cup X)^*$. Let $\Psi : X^* \rightarrow Y^*$ be a morphism. We extend it to

$$\tilde{\Psi} : (V \cup X)^* \rightarrow (V \cup Y)^*$$

by setting

$$\begin{aligned} \tilde{\Psi} : (V \cup X)^* &\rightarrow (V \cup Y)^* \\ x \in X^* &\mapsto \Psi(x) \\ v \in V &\mapsto v \end{aligned} .$$

Take the set $P_{\Psi} = \{v \rightarrow \Psi(w) \mid v \rightarrow w \in P\}$ and define the grammar $\mathcal{G}_{\Psi} = (V, Y, P_{\Psi})$. Then the language $L_{\mathcal{G}_{\Psi}}(v)$ is exactly the language obtained by applying Ψ to $L = L_{\mathcal{G}}(v)$, that is, the language $\Psi(L)$:

$$\begin{aligned} L_{\mathcal{G}_{\Psi}}(v) &= \{\tilde{w} \in Y^* \mid v \xrightarrow{\mathcal{G}_{\Psi}}^* \tilde{w}\} = \{\tilde{w} \in Y^* \mid v \xrightarrow{\mathcal{G}}^* w, \tilde{w} = \Psi(w)\} = \{\Psi(w) \mid v \xrightarrow{\mathcal{G}}^* w\} = \\ &= \{\Psi(w) \mid w \in L_{\mathcal{G}}(v)\}. \end{aligned}$$

Thus, $\Psi(L)$ is a context-free language.

(5). Let $K \subset X^*$ be a regular language with associated deterministic finite state automaton $\mathcal{A} = (Q, X, \delta, I = \{q_0\}, F)$. Consider $q \in Q$ and $t = t_1 t_2 \cdots t_m$, $t_i \in X^*$ for all $i = 1, \dots, m$. We remember the interpretation of δ as a function and denote by $q \cdot t$ the resultant vertex of reading t starting at q (passing through the vertices q_i):

$$q \cdot t = \delta(q, t) = \delta(q_m, t_m) \circ \delta(q_{m-1}, t_{m-1}) \circ \cdots \circ \delta(q_1, t_1).$$

Let σ be a symbol that is not in either X , Q or V and $\mathcal{G} = (V, X, P)$ be the context-free grammar in which $v \in V$ generates L .

We define productions from $\{\sigma\} \cup (Q \times V \times Q)$ to $((\{\sigma\} \cup (Q \times V \times Q)) \cup X)^*$,

$$\tilde{P} = P_1 \cup P_2,$$

$$P_1 = \{\sigma \rightarrow (q_0, u, q) \mid q \in F\}$$

and

$$P_2 = \{(q, v, q') \rightarrow u_0(q_1, \eta_1, q'_1)u_1(q_2, \eta_2, q'_2) \cdots u_{k-1}(q_k, \eta_k, q'_k)u_k \mid k \geq 0, (i), (ii) \text{ and } (iii)\},$$

where

$$(i) \quad v \rightarrow u_0 \eta_1 u_1 \eta_2 \cdots \eta_k u_k \in P;$$

$$(ii) \quad v, \eta_i \in V, u_0, u_i \in X^*, q, q', q_i, q'_i \in Q, \text{ for } 1 \leq i \leq k;$$

(iii) $q \cdot u_0 = q_1$, $q'_i \cdot u_i = q_{i+1}$ for all $1 \leq i \leq k-1$ and $q'_k \cdot u_k = q'$.

Notice that P_2 is finite because of the restriction of taking k 's such that

$$v \rightarrow u_0 \eta_1 u_1 \eta_2 \cdots \eta_k u_k \in P$$

and P is finite. These productions allow us to define the context-free grammar

$$\mathcal{G}_K = (\{\sigma\} \cup (Q \times V \times Q), X, \tilde{P}),$$

such that $L_{\mathcal{G}_K}(q, v, q') = L_{\mathcal{G}}(v) \cap \{t \in X^* \mid q \cdot t = q'\}$ for any $q, q' \in Q, v \in V$. Thus, $L \cap K = L_{\mathcal{G}_K}(\sigma)$ and it is a context-free language.

(7). Let $\Psi : Y^* \rightarrow X^*$ be an alphabetic morphism, which we extend to $\tilde{\Psi}$ as we did in item (2):

$$\begin{aligned} \tilde{\Psi} : (V \cup Y)^* &\rightarrow (V \cup X)^* \\ x \in Y^* &\mapsto \Psi(x) \\ v \in V &\mapsto v \end{aligned} .$$

Set $Z = \{y \in Y \mid \Psi(y) = 1\} \subseteq Y$ and σ a new letter, $\sigma \notin V \cup X \cup Y$. We can define a context-free grammar $\mathcal{G}_{\Psi^{-1}} = (\{\sigma\} \cup V, Y, \tilde{P})$, considering the productions $\tilde{P} = P_1 \cup P_2$:

$$P_1 = \left\{ \sigma \rightarrow 1 + \sum_{z \in Z} \sigma z \right\},$$

where the sum is in the monoid Y and

$$\begin{aligned} P_2 = \{ &v \rightarrow \sigma y_1 \sigma y_2 \sigma \cdots \sigma y_k \sigma \mid k \geq 0 \text{ and} \\ &v \rightarrow \Psi(y_1, y_2, \dots, y_k) \in P, v \in V, y_1, \dots, y_k \in V \cup Z^c \}. \end{aligned}$$

Notice that P_2 is finite because the restriction of Ψ to $(V \cup Z^c)^*$ is strictly alphabetic. Then, $\mathcal{G}_{\Psi^{-1}}$ is well defined and we have $\Psi^{-1}L_{\mathcal{G}}(v) = L_{\mathcal{G}_{\Psi^{-1}}}(v)$ for every $v \in V$, meaning $L_{\mathcal{G}_{\Psi^{-1}}}(\sigma) = Z^*$. Thus, $\Psi^{-1}(L)$ is a context-free language. \square

Though we do not prove it, it is interesting to note that any monoid morphism can have its inverse factorized into an alphabetic morphism intersected with a regular language followed by a morphism. Thus, the above theorem gives us that, in fact, context-free languages are closed under the inverse of any monoid morphism.

Definition 3.1.14. A group G with finite generating set S is a *context-free group* if its language of the word problem $\text{WP}(G, S)$ is context-free.

Remark 3.1.15. In the last definition, as well as on the statement of Anisimov's theorem, we have been neglecting the fact that the word problem language depends on the choice of generating set. We present the proof that the choice of generating set does not change the characteristic of WP being regular, context-free or co-context-free in Chapter 4, Theorem 4.1.3, so that everything we obtain for a specific set S actually holds for any generating set.

Note in the above definition that we do not specify whether the pushdown automaton accepting $WP(G, S)$ is deterministic. We will see, with Muller and Schupp's result, 3.2.12, that in the case of the Word Problem language, being context-free is the same as being deterministic context-free. But this fact is not true for any context-free language. The easiest counter-example makes use of the closure under complementation, which is a property only of the deterministic context-free languages. The proof of this property was taken from Chapter 2, Section 4 of [28].

Proposition 3.1.16. *The class of deterministic context-free languages is closed under complementation.*

Proof. The idea is to do the same that we did in Theorem 2.1.8: invert the accepted and non-accepted states and show that the new automaton accepts the complementary language. The main difference is that a pushdown automaton can keep doing some moves after ending the input string and this could imply accepting words which we do not want to be accepted. We limit when a word is accepted to prevent that.

Consider a deterministic pushdown automaton $\mathcal{M} = (Q, X, \Sigma, \delta, q_0, \perp, F)$ accepting the language \mathcal{L} . We first modify it into a new deterministic pushdown automaton which always reads the entire input string. After that, we do some more changes and finally invert the automaton.

Step 1 | Modifying the automaton by adding new states.

- i. a new start state q_{start} ;
- ii. a new accepted state q_{accept} ;
- iii. a new state q_{reject} ;
- iv. for every $q \in Q$, add a new accepted state q_a .

Define F' as the set of old accepted states, F , together with all the new accepted states. Next, we have to make some changes in the set of transitions.

- i. for every $q \in Q$, if $\delta(q, \varepsilon, b) = (r, y)$, add a new transition $\delta(q_a, \varepsilon, b) = (r_a, y)$;
- ii. for every $q \in Q$ and $x \in X$, if $\delta(q, x, b) = (r, y)$, add a new transition $\delta(q_a, x, b) = (r, y)$;
- iii. for every $q \in F$, change $\delta(q, \varepsilon, b) = (r, y)$ to $\delta(q, \varepsilon, b) = (r_a, y)$;
- iv. add a transition $\delta(q_{start}, \varepsilon, \varepsilon) = (q_0, \$)$;
- v. for every $x \in X$, add a transition $\delta(q_{accept}, x, \varepsilon) = (q_{reject}, \varepsilon)$;

- vi. for every $q \notin F'$ and $x \in X$, add a transition $\delta(q, x, \$) = (q_{reject}, \varepsilon)$;
- vii. for every $q \in F'$ and $x \in X$, add a transition $\delta(q, x, \$) = (q_{accept}, \varepsilon)$;
- viii. for every $x \in X$, add a transition $\delta(q_{reject}, x, \varepsilon) = (q_{reject}, \varepsilon)$.

The transition in *iv* initializes the stack with a new special symbol. If this new symbol is ever detected in a non-accepted state, the automaton goes to the reject state (transitions in *vi*), where it finishes the reading of the input string (transitions in *viii*). If $\$$ is detected in an accepted state, the automaton enters the q_{accept} state (transitions in *vii*) and remains there only if there are no more symbols to be read in the input string (transitions in *v*).

Lastly, we need to make sure that there are no endless sequences of moves preventing the automaton from finishing the input string. Let us call (q, a) a *looping situation* if the automaton enters a state q with a in the top of the stack and after that it never reads an input symbol again, neither does it erase a . It could change states, however. If (q, a) is a looping situation where the automaton eventually enters an accepting state, erase it and add a transition $\delta(q, \varepsilon, a) = (q_{accept}, \varepsilon)$. If (q, a) is a looping situation where the automaton never enters an accepting state, erase it and add a transition $\delta(q, \varepsilon, a) = (q_{reject}, \varepsilon)$.

We now have a new automaton $\widetilde{\mathcal{M}} = (Q', X, \Sigma \cup \{\$\}, \delta, q_{start}, \$, F')$ which always read the input tape to the end, but accepts the same language \mathcal{L} . It also remains in accepted states, once there, until it reads the next input symbol.

Step 2 | Inverting $\widetilde{\mathcal{M}}$ so that it accepts the complementary language.

Before actually changing accepted states into non-accepted states and the opposite as well, we need to identify the states which actually read something from the input string. These are the only states we want as accepted states, in order to prevent the acceptance of words of \mathcal{L} . That is because the automaton $\widetilde{\mathcal{M}}$ could read a word ω and after the reading still do some moves, before it finishes in an accepted state. If some of those moves passed through a non-accepted state, which we now would change into an accepted one, the same word ω would still be accepted by the new automaton.

We call *reading states* the states q for which the automaton reads something from the input tape and does not erase or change anything in the stack, that is, the states such that a transition $\delta(q, x, \varepsilon) = (r, \varepsilon)$ exists for some $r \in Q'$. If there is a state reading the input tape and also doing something to the stack, we need to check if the next reading depends on what is in the stack or not. For that, let us divide such transitions in two steps: for every $x \in X$ and $a \in \Sigma$, if $\delta(q, x, a) = (r, y)$, erase this transition, add a new state q_x and add the transitions $\delta(q, \varepsilon, x) = (q_x, \varepsilon)$ and $\delta(q_x, a, \varepsilon) = (r, y)$. This makes all the q_x into reading states. If $q \in F'$, make q_x also an accepted state, for any $x \in X$ for which it exists. After that, make any accepting state which is not a reading state into a non-accepting state, obtaining a new automaton, which we denote by $\widehat{\mathcal{M}}$ and which also accepts \mathcal{L} , but

only enters accepted states once for each input symbol, at the exact moment when it is about to read the next input.

Finally, we can invert accepted states into non-accepted ones and non-accepted states into accepted ones, obtaining \mathcal{M}^c . The language accepted by \mathcal{M}^c is \mathcal{L}^c .

Step 3 | \mathcal{M}^c accepts \mathcal{L}^c .

First, we notice that the input alphabet has never suffered any change, so that $L(\mathcal{M}^c) \subseteq X^*$. Second, we notice that $\widehat{\mathcal{M}}$ is deterministic: for every word $\omega \in \mathcal{L}$, there is only one path in $\widehat{\mathcal{M}}$ beginning in the start state and ending in an accepted state which is followed by reading ω . We furthermore removed the possibility to reach the accepted state only after finishing to read the input string, or the possibility to leave it after there, so that in \mathcal{M}^c , this same path still is followed by the reading of ω , but it ends in a non-accepted state. Since there is no change in the set of initial states, then there is no other way of reading ω in \mathcal{M}^c , otherwise there would have to be a path in $\widehat{\mathcal{M}}$ which would have to be followed by reading ω , but ending in a non-accepted state, which is impossible because of the deterministic aspect of $\widehat{\mathcal{M}}$. Thus, $\omega \notin L(\mathcal{M}^c)$ so that $\mathcal{L} \cap L(\mathcal{M}^c) = \emptyset$, which means that $L(\mathcal{M}^c) \subseteq X^* \setminus \mathcal{L}$.

Finally, take $\omega \in X^* \setminus \mathcal{L}$. We have built $\widehat{\mathcal{M}}$ in such a way that it always finishes reading the input string, so the question remains if by reading ω , it ends in a non-accepted state. But that is clear, since $\omega \in X^* \setminus \mathcal{L}$. Thus, in \mathcal{M}^c , this end state is an accepted state and ω is accepted. Thus, $X^* \setminus \mathcal{L} \subseteq L(\mathcal{M}^c)$. \square

Let us now see an example of context-free language which does not have context-free complement.

Example 3.1.17. Consider the language $L = \{ww \mid w \in \{0, 1\}^*\}$. Using the so called Pumping Lemma, it can be shown that L is not a context-free language. (For more on this proof, see [28], Section 2.3.) The complement, $M = L^c = \{0, 1\}^* \setminus L$, is however a context-free language. The generating context-free grammar is $\mathcal{G} = (\{S, A, B\}, \{0, 1\}, P)$, where P is the set of productions:

$$\begin{aligned} S &\rightarrow \{A, B, AB, BA\}, \\ A &\rightarrow \{0, 0A0, 0A1, 1A1, 1A0\}, \\ B &\rightarrow \{1, 0B0, 0B1, 1B1, 1B0\}. \end{aligned}$$

So the language M is a context-free language with non-context-free complement, and therefore it cannot be a deterministic context-free language, by the last proposition. Thus, the class of context-free languages is in general strictly greater than the class of deterministic context-free languages.

We finish this section with a partial characterization of context-free groups, a result from [1] stating that abelian context-free groups are exactly the abelian groups with rank

less or equal 1. For that, let us remember that any finitely generated abelian group is isomorphic to the direct product of a finite group by a finite number of copies of the infinite cyclic group \mathbb{Z} . We denote $G \cong T \times \mathbb{Z}^k$, being \mathbb{Z}^k the direct product of k copies of the infinite cyclic group \mathbb{Z} .

Theorem 3.1.18. *A finitely generated abelian group $G \cong T \times \mathbb{Z}^k$ is a context-free group if, and only if, $k \leq 1$.*

We now move on to the next class of languages, which is an intermediate one.

3.1.2 One-Counter Languages

In this section we present an intermediate result connected to the construction of the Chomsky Hierarchy for groups. It consists of a particular case of the context-free languages, which is thoroughly explored in [12].

Definition 3.1.19. *A one-counter pushdown automaton is a 7-uple*

$$\mathcal{M} = (Q, X, \Sigma, \delta, q_0, \$, F)$$

where

1. Q is a finite set, called the set of states of \mathcal{M} ;
2. X is a finite set, called the tape alphabet of \mathcal{M} ;
3. Σ is a two-elements set, called the stack alphabet of \mathcal{M} ;
4. $\delta \subseteq ((Q \cup \{\varepsilon\}) \times (X \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})) \times (Q \times \Sigma^*)$ defines a finite number of transition relations;
5. $q_0 \in Q$ is the initial state;
6. $\$ \in \Sigma$ is called the bottom of the stack;
7. $F \subseteq Q$ is the set of accepted states.

The definition makes it clear that one-counter automata are particular cases of push-down automata, where the stack alphabet has only two elements, the bottom symbol and another one. It is natural to define one-counter groups.

Definition 3.1.20. *A group G with finite generating set S is a one-counter group if its language of the word problem, $\text{WP}(G, S)$, is accepted by a one-counter automaton.*

Let us now state an intermediate result in the classification of groups by automata.

Theorem 3.1.21 (Herbst, [12]). *Consider a group G with finite generating set S . Then $WP(G, S)$ is a one-counter language if, and only if, G is virtually cyclic.*

The proof of this result needs some definitions and other results that are too far from the main goal of this work. Thus, the interested reader is encouraged to consult [12].

3.2 The Theorem of Muller and Schupp

3.2.1 Some Definitions

The theorem by Muller and Schupp characterizing the context-free groups can be obtained from a series of results that require additional definitions. We follow [4] and [10].

Definition 3.2.1. The *involution function* for a directed graph $\Gamma = (V, E, s, t)$ is a function $\bar{\cdot} : E \rightarrow E$ on the directed edges so that for any $e \in E$, we have

- $\bar{\bar{e}} = e \in E$,
- $f \neq e$,
- $s(e) = t(f)$ and
- $t(e) = s(f)$.

Definition 3.2.2. A group G acts on a graph $\Gamma = (V, E)$ if there is an action of G on V , denoted $v \mapsto g \cdot v$ and an action on E , denoted by $e \mapsto g \cdot e$, such that $s(g \cdot e) = g \cdot s(e)$, $t(g \cdot e) = g \cdot t(e)$ and $g \cdot \bar{e} = \overline{g \cdot e}$ for all $g \in G, e \in E$.

Definition 3.2.3. Let G be a group acting on a graph $\Gamma = (V, E)$. The *quotient graph* G/Γ is a graph whose set of vertices is given by the orbits $G \cdot v$, for $v \in V$ and whose set of edges is given by the orbits $G \cdot e$, for $e \in E$.

Definition 3.2.4. Let $\mathcal{G} = (V, X, P)$ be a context-free grammar and $L_{\mathcal{G}}(\alpha)$ be one of its generated languages. Consider the grammar $\tilde{\mathcal{G}}$ which generates $L_{\mathcal{G}}(\alpha)$ only, by adding a start state for \mathcal{G} : $\tilde{\mathcal{G}} = (V, X, P, \alpha)$. Adding a start state means that in the new grammar, the only element from which one can begin the reading of a word, that is, the only element where one can start reading the productions, is the element called start state. The grammar is said to be in *Chomsky normal form* if:

- i. the productions are either of the form $u \xrightarrow[\tilde{\mathcal{G}}]{*} vw$, for $u, v, w \in V$ or of the form $u \xrightarrow[\tilde{\mathcal{G}}]{*} w$ with $u \in V, w \in X$ and $|w| \leq 1$;
- ii. given a production $u \xrightarrow[\tilde{\mathcal{G}}]{*} vw$, then neither v nor w is α ;

- iii. given $u \in V$, there exists a chain of productions such that $\alpha \xrightarrow[\tilde{G}]{}^* u$.

Definition 3.2.5. Given groups G_i , for $i \in I$, generated respectively by X_i and such that for $i \neq j$, $G_i \setminus \{e_{G_i}\} \cap G_j \setminus \{e_{G_j}\} = \emptyset$, the *free product* of $\{G_i\}_{i \in I}$ is $G = \ast_{i \in I} G_i$. G is a group such that

- i. $\langle X_i \rangle = G_i$ is a subgroup of G , for every $i \in I$;
- ii. for each group H with associated set of functions $\{\theta_i : X_i \rightarrow H\}_{i \in I}$ so that θ_i extends to a group homomorphism $\tilde{\theta}_i : G_i \rightarrow H$ for any $i \in I$, there exists a unique group homomorphism $\phi : G \rightarrow H$ respecting

$$\phi|_{X_i} = \theta_i \quad \text{for all } i \in I$$

and that the following diagram must commute, for all $i \in I$

$$\begin{array}{ccc} X_i & \xrightarrow{f} & G \\ & \searrow \theta_i & \downarrow \exists! \phi \\ & & H \end{array}$$

Definition 3.2.6. Given a connected graph $\Gamma = (V, E)$, a *spanning tree* for Γ is a connected subgraph $T = (V_T, E_T)$ such that

- i. $V_T = V$ and
- ii. E_T is the minimal set contained in E which makes the first condition possible.

Definition 3.2.7. A *graph of groups* is a pair $\mathbb{G} = (\mathbb{G}, \Gamma)$ where

- i. $\Gamma = (V, E, s, t)$ is a directed graph with involution function $\bar{\cdot} : E \rightarrow E$.
- ii. $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2)$ is a pair where
 - $\mathbb{G}_1 = \{G_x \mid x \in E \cup V\}$ is a set of groups, which is in bijection with the vertices and edges of Γ ,
 - $\mathbb{G}_2 = \{\alpha_e : G_e \rightarrow G_{s(e)} \mid e \in E\}$ is a set of group homomorphisms associating to each edge group an injective homomorphism into the vertex group at the start of the edge and
 - $G_e = G_{\bar{e}}$ for all $e \in E$.

Definition 3.2.8. Given a graph of groups $\mathcal{G} = (\mathbb{G}, \Gamma)$, with $\Gamma = (V, E, s, t)$, groups G_u , $u \in V \cup E$ and homomorphisms $\alpha_e, e \in E$ as in the definition, the *fundamental group* of the graph of groups \mathcal{G} is denoted by \mathcal{F} .

\mathcal{F} is defined as the fundamental group for Γ with respect to T a spanning tree. We write $\mathcal{F} = \pi_1(\Gamma, T)$ and it is a quotient of the free product $\left(\ast_{v \in V} G_v \right) \ast \left(\ast_{e \in E} G_e \right)$ where the following rules hold:

1. $\bar{e}\alpha_e(g)e = \alpha_{\bar{e}}(g)$ for every $e \in E, g \in G_e$;
2. $e\bar{e} = \bar{e}e = 1$ for all $e \in E$ and
3. $e = 1$ for any $e \in E \cap T$.

Since we consider connected graphs only, it can be shown that the fundamental groups are isomorphic for all spanning trees, so that one can write $\pi_1(\Gamma)$ instead of specifying the considered spanning tree. Now we can define a decomposition over graphs of groups.

Definition 3.2.9. A *splitting* of a group G over a graph of groups $\mathcal{G} = (\mathbb{G}, \Gamma)$ is an isomorphism $\phi : G \rightarrow \pi_1(\Gamma)$. If the edge groups $G_e, e \in E$ are of the class \mathcal{P} of groups, then the splitting is said to be \mathcal{P} .

The last definitions needed before we can go on to the main theorem of this chapter are the tree decomposition and the tree width of a graph.

Definition 3.2.10. A *tree decomposition* of a non-empty connected graph $\Gamma = (V, E, s, t)$ is a pair (T, ϕ) , where $T = (V_T, E_T, ends)$ is an undirected tree,

$$\begin{aligned} \phi : V_T &\rightarrow \mathcal{P}(V) \\ p &\mapsto X_p \end{aligned}$$

is a map and the following conditions are satisfied:

1. $|X_p| < \infty$;
2. for every $v \in V$, there is some $p \in V_T$ such that $v \in X_p$;
3. for every $e \in E$, there is some $p \in V_T$ such that $\{s(e), t(e)\} \subseteq X_p$, and
4. if $v \in X_p \cap X_q$ for some $v \in V$, then for all vertices $r \in V_T$ which are on the geodesic path from p to q , we have $v \in X_r$.

Definition 3.2.11. Consider the notation for a tree decomposition introduced in Definition 3.2.10.

- The sets X_p are called the *bags* of the tree decomposition.

- The *bagsize* of a tree decomposition (T, ϕ) is $\text{bs}(T) = \sup\{|X_p| \mid p \in V_T\}$.
- A connected graph Γ has *finite tree width* if there is some $k \in \mathbb{N}$ for which there exists a tree decomposition (T, ϕ) with $\text{bs}(T) = k$.
- The *tree width* of a connected graph Γ is $p = \inf\{\text{bs}(T) \mid T \text{ is in a tree decomposition of } \Gamma\} - 1$.

3.2.2 Main Results

Theorem 3.2.12 (Muller-Schupp, [22, 23]). *Consider a group G with finite generating set S . The following statements are equivalent:*

1. G is *virtually free*;
2. $\text{WP}(G, S)$ is a *context-free language*;
3. $\text{WP}(G, S)$ is a *deterministic context-free language*.

A recent proof of this theorem can be found in [10]. An annotated version of this proof can be found in [4].

We only state the main results that, put together, prove the previous theorem. Before that, however, we must first make one important note on something we have been neglecting so far. What happens if one changes the generating set S for the group G ? If \tilde{S} is another generating set, then is the condition that $\text{WP}(G, S)$ is context-free enough to assure that $\text{WP}(G, \tilde{S})$ is also context-free? If these assertions were false, then the context-free groups would not even be well defined, so the next result is of much importance.

Theorem 3.2.13. *Let \mathcal{F} be a family of languages and G be a group generated by a finite set S such that $\text{WP}(G, S) \in \mathcal{F}$. If \mathcal{F} is closed under inverse (monoid) homomorphisms, then $\text{WP}(G, K) \in \mathcal{F}$ for every finite generating set K of G .*

We present the proof of a more general version of Theorem 3.2.13 later in Theorem 4.1.3. Thus, since closure under inverse homomorphisms was proven for the context-free languages in 3.1.13 and its following observation, we can proceed without worries. The same type of closure holds for regular languages as well.

Theorem 3.2.14. *Suppose $G = \langle S \mid R \rangle$ is a finitely generated group whose language of the word problem $\text{WP}(G, S)$ is context-free. Then G has a locally finite Cayley graph $\Gamma(G, S)$ of finite tree width.*

Theorem 3.2.15. *Suppose $G = \langle S \mid R \rangle$ is a finitely generated group whose Cayley graph $\Gamma(G, S)$ is locally finite and is of finite tree width. Then G splits over a graph of groups with finite vertex groups and finite edge groups.*

Theorem 3.2.16. *Suppose $\mathcal{G} = (\mathbb{G}, \Gamma)$ is a graph of groups, where Γ is connected and every vertex group is finite. If T is a spanning tree in Γ , then $\pi_1(\mathcal{G}, T) = \pi_1(\Gamma, T)$ is a virtually free group.*

The last three results imply the more difficult direction of the claim in 3.2.12. The other direction can be shown directly, but we omit the proof.

4. Co-Context-Free Groups

In this chapter we introduce co-context-free groups, which are investigated as the possible next step in Chomsky's Hierarchy for groups in [15]. We go through some closure properties and mention other interesting known properties of this class of groups.

The definitions and results presented in the sections are all taken from [11] and [15].

4.1 Finitely Generated Subgroups

Definition 4.1.1. Consider a group G with finite generating set S . The *language of the co-word problem* for G is $\text{coWP}(G, S) = \text{WP}(G, S)^c \subseteq (S \cup S^{-1})^*$, which is the set of nontrivial elements of G .

This definition was not made in the case of regular languages because if L is a regular language in the alphabet S , then so is $L^c \subseteq \overline{S}^*$, so that the study of $\text{WP}(G, S)$ suffices to obtain interesting results.

Definition 4.1.2. A group is called *co-context-free* if its co-word problem language is a context-free language.

We should worry about the fact that the co-word problem is defined in dependence of the generating set, so let us prove Theorem 3.2.13 and extend it.

Theorem 4.1.3. *Given \mathcal{F} a family of languages closed under inverse (monoid) homomorphisms and G a finitely generated group, then*

1. *if S is a generating set such that $\text{WP}(G, S) \in \mathcal{F}$, then $\text{WP}(G, K) \in \mathcal{F}$ for every finite generating set K of G .*
2. *if S is a generating set such that $\text{coWP}(G, S) \in \mathcal{F}$, then $\text{coWP}(G, K) \in \mathcal{F}$ for every finite generating set K of G .*

Proof. Let S, K be two finite generating sets for G . There is a natural homomorphism

$$\phi : (S \cup S^{-1})^* \rightarrow (K \cup K^{-1})^*$$

where each word $\omega \in (S \cup S^{-1})^*$ is taken into a word $\tilde{\omega} \in (K \cup K^{-1})^*$ that represents the same element $g \in G$ represented by ω . We induce this homomorphism by first expressing

$s \in S \cup S^{-1}$ as an element of G and finding a corresponding word in $(K \cup K^{-1})^*$. That is, if

$$f_1 : (S \cup S^{-1})^* \rightarrow G$$

$$f_2 : (K \cup K^{-1})^* \rightarrow G$$

are the natural defined functions given by the universal property between the set of words on the generating sets and the group (which is a free monoid, therefore having universal property), then for $\omega \in (S \cup S^{-1})^*$ with $f_1(\omega) = g$, we have that $f_2 \circ \phi(\omega) = g$ and the following diagram commutes

$$\begin{array}{ccc} (S \cup S^{-1})^* & \xrightarrow{\phi} & (K \cup K^{-1})^* \\ & \searrow f_1 & \swarrow f_2 \\ & G & \end{array} .$$

1. We have that $\text{WP}(G, S) = \phi^{-1}(\text{WP}(G, K))$, so that if the property of being \mathcal{F} is closed under inverse homomorphism (as it is) and $\text{WP}(G, S)$ is \mathcal{F} , $\text{WP}(G, K)$ is also \mathcal{F} .
2. We have that $\text{coWP}(G, S) = \phi^{-1}(\text{coWP}(G, K))$, so that if being a \mathcal{F} language is closed under inverse homomorphism (as it is) and $\text{coWP}(G, S)$ is \mathcal{F} , $\text{coWP}(G, K)$ is also \mathcal{F} . □

Theorem 4.1.3 guarantees that all of the so far discussed results and definitions were well defined, even if we have neglected the dependency on the generating set of the definition of the word problem language of a finitely generated group. Let us make some notation conventions.

Notation 4.1.4.

- i. Given an alphabet S , we continue using notation 1.2.23 and denote by \bar{S} the set $(S \cup S^{-1})$.
- ii. Given a finitely generated group G , we say that G is a \mathcal{F} -group if $\text{WP}(G, S)$ has property \mathcal{F} for all finite generating sets S .
- iii. Given a finitely generated group G , we say that G is a co- \mathcal{F} -group if $\text{coWP}(G, S)$ has property \mathcal{F} for all finite generating sets S .
- iv. Given a group G , its identity element $e = e_G$ is also denoted by 1_G .

Before proving some closure properties for co-context-free groups, we should understand why they are considered to be the next class in the construction of Chomsky's hierarchy for groups.

Theorem 4.1.5. *Let G be a context-free group. Then G is co-context-free.*

The proof is immediate, putting together results 3.1.16 and 3.2.12. This fact makes it clear that co-context-free groups are a larger (or equal) class than the class of context-free groups. However, the Muller & Schupp theorem, together with the closure under finite direct product, which is proved in Corollary 4.2.10 of Section 4.2, ensures the existence of groups which are co-context-free but not context-free. As an example, take any finitely generated non-cyclic free abelian group G , which is a finite direct product of infinite cyclic groups. By Theorems 3.1.21 and 4.1.5, G is the finite direct product of co-context-free groups and is therefore a co-context-free group. But G is not context-free, since it violates Muller & Schupp's result by not being virtually free.

We go on for the closure properties, keeping in mind the results from Theorem 3.1.13.

Theorem 4.1.6 (Holt-Rees-Röver-Thomas, [15]). *Let \mathcal{F} be a class of languages closed under inverse homomorphisms and intersection with regular sets. Then \mathcal{F} -groups and co- \mathcal{F} -groups are closed under taking finitely generated subgroups.*

Proof. Take a finitely generated group G and a finitely generated subgroup of G , H . Choose S_H a finite generating set of H and extend it to a finite generating set S of G . Note that this can always be done, since one could simply take any finite generating set K of G and consider $S = K \cup S_H$.

We have that $\text{WP}(H, S_H) = \overline{S_H}^* \cap \text{WP}(G, S)$:

\subseteq Take $\omega \in \text{WP}(H, S_H)$, $\omega = a_1 \cdots a_n$, $a_i \in S_H$ for any $1 \leq i \leq n$. Now, $\omega = e_H = e_G$ because H is a subgroup. Then, since $S_H \subseteq S$, $\omega = a_1 \cdots a_n$, $a_i \in S$ for any $1 \leq i \leq n$, and $\omega \in \text{WP}(G, S)$. Besides that, $\text{WP}(H, S_H) \subseteq \overline{S_H}^*$. Thus, $\omega \in \overline{S_H}^* \cap \text{WP}(G, S)$.

\supseteq Take $\omega \in \overline{S_H}^* \cap \text{WP}(G, S)$. Since $\omega \in \text{WP}(G, S)$, we have $\omega = b_a \cdots b_m = e_G$, with $b_i \in S$. But $\omega \in \overline{S_H}^*$ too, so that $\omega = a_1 \cdots a_n$, $a_i \in S_H$ for any $1 \leq i \leq n$. Then, we have $a_1 \cdots a_n = e_G = e_H$ and $\omega \in \text{WP}(H, S_H)$, by definition.

Analogously, $\text{coWP}(H, S_H) = \overline{S_H}^* \cap \text{coWP}(G, s)$. If we prove that $\overline{S_H}^*$ is a regular language in \overline{S}^* , then we have that $\text{WP}(H, S_H)$ (respectively, $\text{coWP}(H, S_H)$) is the intersection of a \mathcal{F} language with a regular one, thus being also a \mathcal{F} language, by hypothesis. The closure under inverse homomorphisms completes the proof for independence of chosen finite generating set.

But $\overline{S_H}^*$ is trivially a regular language, as is any "all-strings" set on finite alphabets. Let us show that by constructing a deterministic finite state automaton that accepts $\overline{S_H}^*$, see Figure 4.1. Since S_H is finite, we can write $S_H = \{s_1, s_2, \dots, s_n\}$. Consider two circles in a state to identify it as an accepting state and a star \star inside a state to identify it as the start state. We use the star in order to have no confusion with the set of generators S of G , although in literature the start state is usually denoted with a S inside it.

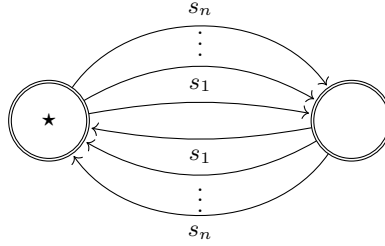


Figure 4.1 – Automaton accepting $\overline{S_H}^*$.

The automaton above accepts $\overline{S_H}^*$ and is a finite state (deterministic) automaton, so that the language is regular as we desired. \square

4.2 Finite Direct Products

Let us define the *shuffle* of two languages. Intuitively, it is a set obtained by interwining the elements of two other sets. We define that set and after that show how to obtain it from automata. These definitions are helpful for the proof of closure under taking direct products.

Definition 4.2.1. Consider two languages $L_1 \subseteq \Sigma^*$ and $L_2 \subseteq \Delta^*$. The *shuffle* of L_1 with L_2 is

$$L_1 \leftrightarrow L_2 = \{x_1y_1 \cdots x_ny_n \mid x_1x_2 \cdots x_n \in L_1, y_1y_2 \cdots y_n \in L_2, x_i \in \Sigma^*, y_i \in \Delta^*\}.$$

Definition 4.2.2. A *generalised sequential machine* (gsm) is a 6-tuple $\mathcal{M} = (Q, X, \Sigma, \delta, \lambda, q_0)$, where

1. Q is the finite non-empty set of states;
2. X is the alphabet of inputs;
3. Σ is the alphabet of outputs;
4. $\delta : (Q \times X) \rightarrow Q$ is the next-state function;
5. $\lambda : (Q \times X) \rightarrow \Sigma^*$ is the output function and

6. $q_0 \in Q$ is the start state.

\mathcal{M} is said to be a *complete sequential machine* if $\lambda : (Q \times X) \rightarrow \Sigma$.

The generalised sequential machines, instead of accepting/generating languages as we have been seeing until here, generate functions, called generalised sequential machine mappings. Particularly, if one of the languages L_1 or L_2 is regular, say L_2 , then the shuffle is the image of L_1 under a generalised sequential machine mapping.

Definition 4.2.3. Consider a generalised sequential machine $\mathcal{M} = (Q, X, \Sigma, \delta, \lambda, q_0)$. Extend the functions δ and λ to $Q \times X^*$ by induction with

- $\delta(q, \epsilon) = q$,
- $\lambda(q, \epsilon) = \epsilon$,
- $\delta(q, xy) = \delta[\delta(q, x), y]$ and
- $\lambda(q, xy) = \lambda(q, x)\lambda[\delta(q, x), y]$,

where $q \in Q, x \in X^*$ and $y \in X$. Then the operation defined by $\mathcal{M}(x) = \lambda(q_0, x)$, for each $x \in X^*$, is called a *generalised sequential machine mapping*.

Example 4.2.4. In order to help us understand what the function generated by \mathcal{M} is, let us represent it as an automaton, where we write $a|b$ over an edge to indicate that the input symbol being a allows movement along this edge and the output string is then added of b .

The states correspond to Q . We use accepted states to confirm the validity of the input, although it is important to remember we do not generate languages: the gsm generates a function, defining both its domain and image. The domain is the set of words which are accepted by the automaton. The image of each element in the domain is given by the output string. Notice that the existence of edges is related to the definition of δ and the labels are correlated to the definition of λ .

Thus, the automaton in Figure 4.2 is a generalised sequential machine whose map takes any word $\omega \in \{a, b\}^*$ to a word x^n , being n the number of a 's in ω .

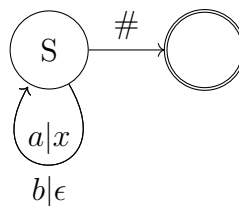


Figure 4.2 – A Generalised Sequential Machine

The symbol $\#$ is what the machine interprets as reading when the input word is finished.

Theorem 4.2.5 (Holt-Rees-Röver-Thomas, [15]). *Let \mathcal{F} be a class of languages closed under shuffle with regular languages and closed under finite union. Then the class of co- \mathcal{F} -groups is closed under taking finite direct products.*

Proof. Consider finitely generated groups G_1 and G_2 , generated respectively by S_1 and S_2 . Assume $\text{coWP}(G_i, S_i) \in \mathcal{F}$, for $i = 1, 2$ and \mathcal{F} as in the statement of the theorem. The direct product is $G = G_1 \times G_2$, which is most certainly generated by $S = (S_1 \times \{1\}) \cup (\{1\} \times S_2)$. We identify S_1 with $S_1 \times \{1\}$ and S_2 with $\{1\} \times S_2$ in G . Now, if

$$\text{coWP}(G, S) = (\text{coWP}(G_1, S_1) \leftrightarrow \overline{S_2^*}) \cup (\text{coWP}(G_2, S_2) \leftrightarrow \overline{S_1^*}),$$

then we have finished the proof by our hypotheses. Let us prove this equality.

First, we need to rewrite an element of G , $(g_1, g_2) \in G$, as a word in $\overline{S^*}$. For that, write g_1 and g_2 in terms of the corresponding generating sets and take the rule of intertwining the entries: $(g_1, g_2) = (a_1 \cdots a_n, b_1 \cdots b_m)$, with $a_i \in S_1$ and $b_j \in S_2$ for $1 \leq i \leq n$, $1 \leq j \leq m$, $n \leq m$, becomes $(g_1, g_2) = a_1 b_1 a_2 b_2 \cdots a_n b_n b_{n+1} \cdots b_m$. If $n \geq m$, just add the extra terms of the first group at the end of the word, exactly as done in the formula above for the case $m \geq n$.

\subseteq Take $\omega \in \text{coWP}(G, S)$. This means, if $\omega = (\omega_1, \omega_2)$, that either $\omega_1 \neq 1_{G_1}$ or $\omega_2 \neq 1_{G_2}$. Without loss of generality, let us assume $\omega_1 \neq 1_{G_1}$. As explained above, we can write $\omega = a_1 b_1 a_2 b_2 \cdots a_n b_n b_{n+1} \cdots b_m$, where we could change the extra b_j 's for extra a_i 's if ω_1 happens to be written with more terms than ω_2 . Since this change would need a completely analogous proof, we consider the case $n \leq m$ only.

That means we have $\omega = a_1 b_1 a_2 b_2 \cdots a_n b_n \cdot b_{n+1} s \cdot s^{-1} b_{n+2} s \cdot s^{-1} \cdots b_m$, for some $s \in S_1$, meaning that $s \cdot s^{-1} \in \overline{S_1^*}$. Thus, since $\omega_1 \neq 1_{G_1}$, we have

$$\omega_1 \cdot s s^{-1} s s^{-1} \cdots s s^{-1} \neq 1_{G_1}$$

and then

$$\omega \in (\text{coWP}(G_1, S_1) \leftrightarrow \overline{S_2^*}) \subseteq (\text{coWP}(G_1, S_1) \leftrightarrow \overline{S_2^*}) \cup (\text{coWP}(G_2, S_2) \leftrightarrow \overline{S_1^*}).$$

Note that $b_{n+1} b_{n+2} \cdots b_m$ could be equal the identity in G_2 , but in that case, we could pick $l \in S_1$ and write $\omega = a_1 b_1 a_2 b_2 \cdots a_n b_n l \cdot l^{-1} b_{n+1} \cdot b_{n+2} s \cdot s^{-1} \cdots b_m s \cdot s^{-1}$, such that $b_{n+2} \cdots b_m \neq 1_{G_2}$.

\supseteq Given

$$\omega \in (\text{coWP}(G_1, S_1) \leftrightarrow \overline{S_2^*}) \cup (\text{coWP}(G_2, S_2) \leftrightarrow \overline{S_1^*}),$$

consider $\omega \in (\text{coWP}(G_1, S_1) \leftrightarrow \overline{S_2}^*)$ without loss of generality. Then,

$$\omega = a_1 b_1 \cdots a_n b_n,$$

with $a_i \in \text{coWP}(G_1, S_1)$ and $b_j \in \overline{S_2}^*$, $1 \leq i, j \leq n$. Since, by construction, ω is the element $(a_1 \cdots a_n, b_1 \cdots b_n) = g \in G$ and $a_1 \cdots a_n \neq 1_{G_1}$, then $g \neq (1_{G_1}, 1_{G_2})$ and $g \in \text{coWP}(G, S)$. Since $g = \omega$, the proof is done. \square

It remains to be seen that the context-free languages are closed under shuffles, which is the same as being closed under generalised sequential machines mappings. From here on, we abbreviate generalised sequential machines as *gsm*. Let us first introduce an auxiliary automaton.

Definition 4.2.6. A *sequential transducer* is a 5-uple $\mathcal{M} = (Q, X, \Sigma, H, q_0)$, where

1. Q is the finite non-empty set of states;
2. X is the alphabet of inputs;
3. Σ is the alphabet of outputs;
4. $H \subseteq Q \times X^* \times \Sigma^* \times Q$ is the next-state finite set and
5. $q_0 \in Q$ is the start state.

An element $(p, u, v, q) \in H$ denotes that the input word u at the state p results in an output word v and that the next state is q . One could think of H as a compression of the gsm functions δ and λ into only one item.

Definition 4.2.7. Given a sequential transducer $\mathcal{M} = (Q, X, \Sigma, H, q_0)$, for each $u \in X^*$, define

$$\mathcal{M}(u) = \{v = v_1 \cdots v_k \mid v_i \in \Sigma^*, \exists \{u_i\}_{i=1}^k \subseteq X^*, \{q_i\}_{i=1}^k \subseteq Q \text{ such that } (q_{i-1}, u_i, v_i, q_i) \in H \text{ for } 1 \leq i \leq k \text{ and } u = u_1 \cdots u_k\}.$$

Then the function \mathcal{M} such that $\mathcal{M}(U) = \bigcup_{u \in U} \mathcal{M}(u)$ is a *sequential transducer mapping* for each $U \subseteq X^*$.

Theorem 4.2.8. Consider a gsm $\mathcal{N} = (Q, X, \Sigma, \delta, \lambda, q_0)$. Then the gsm mappings arising from \mathcal{N} preserve context-free languages.

Proof. Step 1 We first need to notice that each gsm mapping (Definition 4.2.3) is equivalent to a sequential transducer mapping. Thus, we can prove the result for sequential transducer mappings and it holds for the gsm mappings. Our gsm \mathcal{N} is the sequential transducer $\mathcal{M} = (Q, X, \Sigma, H, q_0)$, being H the set $\{(p, a, a, p) \mid p \in Q\} \cup \{(p, u, \lambda(p, u), \delta(p, u)) \mid (p, u) \in$

$Q \times X$ }. Notice that the first set in the union takes into account the first two conditions in the definition of gsm's and the second set in the union refers to the last two conditions in the definition of gsm's. Therefore, we just change the way in which the automaton processes the word, while preserving the actions regarding inputs and outputs, so that both machines are truly generating the same maps.

Step 2 Take L a context-free language. Since we want to take its image under the gsm mapping, we have to consider $L \subseteq X^*$. We construct a morphism θ and a set B such that $\theta(B) = \mathcal{N}(L) = \mathcal{M}(L)$. Then, it remains to be proved that B is a context-free language, because we already know that morphisms preserve context-free languages from Theorem 3.1.13.

Consider a sequential transducer $\mathcal{M}' = (Q', X, \Sigma', H', q_0)$, where $\Sigma' = \Sigma \cup \{z_0\}$, $z_0 \notin \Sigma$ being a new symbol; H' consists of all the 4-uples in one of the forms:

- i. $(p, u, v, q) \in H$ with $|u| \leq 1$ or
- ii. $(p, u_1, z_0, t_1^r), (t_1^r, u_2, z_0, t_2^r) \cdots, (t_{k(r)-2}^r, u_{k(r)-1}, z_0, t_{k(r)-1}^r)$ and $(t_{k(r)-1}^r, u_{k(r)}, v, q)$, for each $r = (p, u, v, q) \in H$ such that $u = u_1 \cdots u_{k(r)} \in X^*$ and $k(r) = |u|$, being $t_1^r, \cdots, t_{k(r)}^r$ abstract symbols;

and $Q \cup \{t_j^r\}_{r \in H, 1 \leq j \leq k(r)} \subseteq Q'$.

We can now consider the set $A \subseteq H'^*$ consisting of all words of the form

$$(q_0, \omega_1, y_1, q_1)(q_1, \omega_2, y_2, q_2) \cdots (q_{n-1}, \omega_n, y_n, q_n)$$

with each $(q_{i-1}, \omega_i, y_i, q_i) \in H'$, $1 \leq i \leq n$, and $y_n \neq z_0$. We need only y_n to be different from z_0 , because the transitions should give outputs in the original alphabet. That is the reason why we defined H' to give as outputs only words that the previous transducer produced.

We can also consider the set A_L , for each $L \subseteq X^*$,

$$A_L = \{(q_0, \omega_1, y_1, q_1) \cdots (q_{m-1}, \omega_m, y_m, q_m) \mid (q_{i-1}, \omega_i, y_i, q_i) \in H' \text{ and } \omega_i \in L \text{ for } 1 \leq i \leq m\}.$$

Notice that the set A_L can be constructed for any language in the alphabet of the gsm, but since we want to preserve context-free languages, we have by hypothesis that L is a context-free language for all our further work in this proof.

Take $B = A_L \cap A$. Construct the morphism

$$\begin{aligned} \theta : \quad H'^* &\rightarrow \Sigma'^* \\ (p, \omega, y \neq z_0, q) &\mapsto y \\ (p, \omega, z_0, q) &\mapsto \epsilon. \end{aligned}$$

For each fixed L , we have that $\theta(B) = \mathcal{M}(L)$ by construction. Also, θ is a substitution. (In fact, it is a semigroup homomorphism, but we do not need that.)

Step 3 To finish this proof, we need to show that B is a context-free language. After that, the result come from the closure under morphisms of the context-free languages, Theorem 3.1.13.

What we do is define a substitution, which preserves context-free languages, such that its image, maybe united with some set, intersected to the set A is exactly B . Then, if we can prove that A is regular, since context-free languages are closed under intersection with regular languages, we have finished the proof.

Define the substitution $\phi : X \rightarrow H'^*$ by

$$\phi(u) = \{\chi_1(p, \omega, y, q)\chi_2 \mid \chi_1, \chi_2 \in \{(p, \epsilon, y, q)\}^* \subseteq H'^* \text{ and } (p, \omega, y, q) \in H'\}.$$

Consider a word in B . It has to be of the form

$$(q_0, \omega_1, y_1, q_1) \cdots (q_{m-1}, \omega_m, y_m, q_m), \text{ each } (q_{i-1}, \omega_i, y_i, q_i) \in H', \omega_i \in L \text{ and } y_n \neq z_0. \quad (4.1)$$

If L does not contain ϵ , then $\nu \in \phi(L) \cap A$ has exactly the form (4.1). Hence,

$$B = \phi(L) \cap A.$$

If L contains ϵ , then $\nu \in B$ with form (4.1) and it is in

$$R_L = (\phi(L) \cup \{(p, \epsilon, y, q) \mid (p, \epsilon, y, q) \in H'\}^*) \cap A.$$

Notice that $\phi(L)$ and $\{(p, \epsilon, y, q) \mid (p, \epsilon, y, q) \in H'\}^*$ are context-free, the first because of Theorem 2.1.8 and because each $\phi(u)$ is the concatenation of H'^* with $H'H'^*$; and the second because the set of all words is regular (see the proof of Theorem 4.1.6) and therefore context-free. Thus we have the intersection of a context-free language with A . Hence, $B = D \cap A$, with D context-free.

Now, consider the set $A^c = H'^* \setminus A$. By construction of A , we have that A^c is the union of all sets having one of the forms:

- i. $\{\epsilon\}$;
- ii. $(p, \omega, y, q)\chi$, where $(p, \omega, y, q) \in H'$, $p \neq q_0$ and $\chi \in H'^*$;
- iii. $\chi_1(p, \omega, y, q)(q', \omega', y', q'')\chi_2$, where $(p, \omega, y, q), (q', \omega', y', q'') \in H'$, $q \neq q'$ and $\chi_1, \chi_2 \in H'^*$; and
- iv. $\chi(p, \omega, z_0, q)$, being $\chi \in H'^*$.

But all these sets are regular, by Theorem 2.1.8, and finite union of regular sets is still a regular set. Hence, A^c is regular. Since regular languages are closed under complementation, then A is regular. Thus, since $\phi(L)$ is context-free, $B = \phi(L) \cap A$ is context-free, and so is $B = D \cap A$. \square

In fact, the context-free languages are also preserved by inverse gsm mappings.

Corollary 4.2.9. *Given a gsm $\mathcal{M} = (Q, X, \Sigma, \delta, \lambda, q_0)$, if \mathcal{M}^{-1} is the operation taking each $A \subseteq \Sigma^*$ into $\mathcal{M}^{-1} = \{\omega \mid \mathcal{M}(\omega) \in A\}$, then \mathcal{M}^{-1} preserves context-free languages.*

Proof. Consider a sequential transducer $\mathcal{N} = (Q, \Sigma, X, H, q_0)$ where H consists of all 4-uples in one of the forms:

- i. $(p, \epsilon, \epsilon, p)$, for all $p \in Q$;
- ii. $(p, \lambda(p, x), x, \delta(p, x))$, for all $(p, x) \in Q \times X$.

Let us show that $\mathcal{M}^{-1}(L) = \mathcal{N}(L)$, for every $L \subseteq \Sigma^*$, so that the result follows from the previous proof.

\subseteq Take $L \subseteq \Sigma^*$ and $x \in \mathcal{M}^{-1}(L) \subseteq X^*$. That means that there is a $u \in L$ such that $\mathcal{M}(x) = \lambda(q_0, x) = u$. We want x to be in $\mathcal{N}(L)$. Take the 4-uple $(q_0, \lambda(q_0, x), x, \delta(q_0, x)) = h$ and observe that, by construction, $h \in H$. That means that

$$\begin{aligned} x \in \{v \mid (q, \lambda(q, v), v, \delta(q, v)) \in H, q \in Q\} &\subseteq \{v = v_1 \cdots v_k \mid v_i \in X^*, \exists \{u_i\}_{i=1}^k \subseteq \Sigma^*, \\ &\quad \{q_i\}_{i=1}^k \subseteq Q \text{ such that } (q_{i-1}, u_i, v_i, q_i) \in H, 1 \leq i \leq k \text{ and } l = u_1 \cdots u_k\} = \\ &= \mathcal{N}(l) \subseteq \mathcal{N}(L), \text{ for some } l \in L. \end{aligned}$$

\supseteq Take $L \subseteq \Sigma^*$ and $x \in \mathcal{N}(L) \subseteq X^*$. This means that

$$\begin{aligned} x \in \{v \mid (q, \lambda(q, v), v, \delta(q, v)) \in H, q \in Q\} &\subseteq \\ &\subseteq \{v = v_1 \cdots v_k \mid v_i \in X^*, \exists \{u_i\}_{i=1}^k \subseteq \Sigma^*, \{q_i\}_{i=1}^k \subseteq Q \text{ such that } (q_{i-1}, u_i, v_i, q_i) \in H, \\ &\quad 1 \leq i \leq k \text{ and } l = u_1 \cdots u_k\} \end{aligned}$$

for some $l \in L$. Now, since the 4-uples in H are specific, we have that either $(q_{i-1}, u_i, v_i, q_i) = (q, \epsilon, \epsilon, q)$ or $(q_{i-1}, u_i, v_i, q_i) = (q, \lambda(q, p), p, \delta(q, p))$, where we have already changed v for x .

In the first case, $x = \epsilon$, which implies $x \in \mathcal{M}^{-1}(L)$, as any language must have the empty word.

In the second case, we have $x_i = p$ and then $\lambda(q_{i-1}, x_i) = u_i$ and $\delta(q_{i-1}, x_i) = q_i$. Remembering the extension of λ and δ to $Q \times X^*$, we have

$$\begin{aligned}\lambda(q_0, x) &= \lambda(q_0, x_1 \cdots x_k) = \lambda(q_0, x_1)\lambda[\delta(q_0, x_1), x_2 \cdots x_k] \\ &= u_1\lambda(q_1, x_2 \cdots x_k) = \cdots = u_1u_2 \cdots u_k = l \in L.\end{aligned}$$

Thus, $\mathcal{M}(x) = \lambda(q_0, x) \in L$, meaning that $x \in \mathcal{M}^{-1}(L)$. \square

Finally, the next corollary follows from Theorems 4.2.5 and 4.2.8.

Corollary 4.2.10. *The class of co-context-free groups is closed under taking finite direct products.*

4.3 Finite Index Overgroups

Definition 4.3.1. Let G be a group and $H \subseteq G$ be a finite index subgroup of G . We say that G is a finite index *overgroup* of H .

Theorem 4.3.2 (Holt-Rees-Röver-Thomas, [15]). *Let \mathcal{F} be a family of languages closed under union with regular sets and inverse generalised sequential machine mappings. Then the class of \mathcal{F} -groups and the class of co- \mathcal{F} -groups are closed under passing to finite index overgroups.*

Before proving the result, let us notice that, since every regular language is a context-free language, context-free languages are, indeed, closed under union with regular languages. This fact, together with Corollary 4.2.9 in the previous section, guarantees that Theorem 4.3.2 applies to context-free languages.

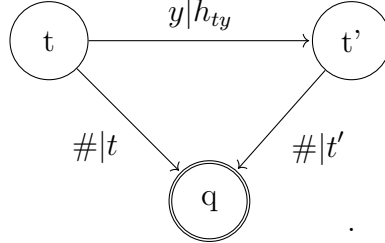
Proof. Let G be a finitely generated group and H be a finite index subgroup of G so that H is a (co-) \mathcal{F} -group. Consider a right transversal T for H in G , with $1 \in T$, so that every element $g \in G$ can be written as $g = ht$, for some $h \in H$. Take a finite generating set X for H and consider $Y = X \cup (T \setminus \{1\})$. Clearly, Y is a finite generating set for G , because both X and T are finite and $G = \bigcup_{t \in T} Ht$.

We define, for each $y \in \bar{Y}$ (remember Notation 1.2.23) and $t \in T$, the word $h_{ty} \in \bar{X}^*$ such that $ty =_G h_{ty}t'$ for some $t' \in T$. We can then construct a generalised sequential machine $\mathcal{M} = (T \cup \{q\}, \bar{Y}, \bar{X}, \delta, \lambda, 1 \in T)$, where

1. $q \notin T$;
2. $\delta(t, y) = t'$, $\delta(t, \#) = q$ and $\delta(t', \#) = q$, being $\#$ the symbol for the end of the input and $t, t' \in T$; and

3. $\lambda(t, y) = h_{ty}$, $\lambda(t', \#) = t'$ and $\lambda(t, \#) = t$, being $\#$ the symbol for the end of the input and $t, t' \in T$.

In order to help us understand what the function generated by \mathcal{M} is, let us represent it as an automaton with start state $1 \in T$:



Looking at the automaton, it is easy to see what our gsm mapping does: given $\omega \in \bar{Y}^*$, it returns the output word $\omega't$ with $t \in T$, $\omega' \in \bar{X}^*$ such that $\omega =_G \omega't$. In order to make it perfectly clear, let us take an example word $\omega = y_1y_2y_3$ with $y_i \in \bar{Y}$. We start in the state 1, by construction. In the figure, this is the same as being in the state t , choosing $t = 1$. Hence, we have to add h_{1y_1} to our output, and travel to the state t' . Note that, since $t' \in T$ and we are not at the end of the input string, it is as if we were in the state t of the figure, only now labeled with t' . We read the input y_2 , so that we have to add $h_{t'y_2}$ to the output and travel to the state t'' such that $t'y_2 =_G h_{t'y_2}t''$. Again we do the same with y_3 and our output becomes $h_{1y_1}h_{t'y_2}h_{t''y_3}t'''$. Hence, we have $\omega' = h_{1y_1}h_{t'y_2}h_{t''y_3} \in \bar{X}^*$ and $t = t''' \in T$, with

$$\begin{aligned} \omega't &= h_{1y_1}h_{t'y_2}h_{t''y_3}t''' =_G h_{1y_1}h_{t'y_2}t''y_3 =_G h_{1y_1}t'y_2y_3 \\ &= 1y_1y_2y_3 = \omega. \end{aligned}$$

We end the proof by showing that

$$\text{WP}(G, Y) = \phi^{-1}(\text{WP}(H, X))$$

and

$$\text{coWP}(G, Y) = \phi^{-1}(\text{coWP}(H, X) \cup A),$$

where ϕ is the gsm mapping we constructed and $A = \{\omega t \mid \omega \in \bar{X}^*, t \in T \setminus \{1\}\}$. These equalities finish the proof because we have \mathcal{F} -closure under union with regular sets and inverse gsm mappings by hypothesis, and A is a regular set by 2.1.8, as it is the concatenation of a set of all strings (which is regular) and a set with finite number of elements (which is also regular).

$$\text{WP}(G, Y) = \phi^{-1}(\text{WP}(H, X))$$

⊆ Take $\omega \in \text{WP}(G, Y)$. We have that $1_G = \omega =_G \phi(\omega) = \omega't$. That implies $\omega't =_H 1_H$, which in turn implies $\omega' = t^{-1}$. But, by construction, $\omega' \in \bar{X}^*$, which

means $\omega' \in H$. Since H is a group, this implies $t \in H$ and since $1 \in T$, we have $t = 1$. Thus, $\phi(\omega) = \omega' =_H 1_H$.

\supseteq Take $\omega \in \phi^{-1}(\text{WP}(H, X))$, then there exists $\tilde{\omega} \in \text{WP}(H, X)$ such that $\phi(\omega) = \tilde{\omega}$. But we know that $\phi(\omega) = \omega't$, with $\omega' \in \overline{X}^*$ and $t \in T$. We have $1 =_H \tilde{\omega} = \omega't$, $\omega' \in H$, $\omega't \in Ht$. Since we have $1 \in T$ and $1 \in Ht$, $t \in T$, t must be 1. (If $t \neq 1$, then there is $h \in H$ such that $ht = 1$, because $1 \in Ht$. But then we have $1 \in Ht$ and $1 \in H = H1$, so that both t and 1 are elements for the same coset in the same transversal, which contradicts the definition of transversals.) Hence, we have $\omega =_G \omega't = \omega' = 1_H = 1_G$.

$$\text{coWP}(G, Y) = \phi^{-1}(\text{coWP}(H, X) \cup A)$$

\subseteq Take $\omega \in \text{coWP}(G, Y)$. We have $1_G \neq \omega =_G \phi(\omega) = \omega't$ for some $\omega' \in \overline{X}^*$, $t \in T$. Thus, $1_H =_H 1_G \neq \omega't$. If $t = 1$, it follows immediately that $\omega' \in \text{coWP}(H, X)$ and since $\omega' = \phi(\omega)$, we have $\omega \in \phi^{-1}(\text{coWP}(H, X) \cup A)$.

If $t \neq 1$, we have by definition of A that $\omega't \in A$, and then

$$\omega \in \phi^{-1}(\text{coWP}(H, X) \cup A).$$

\supseteq Take $\omega \in \phi^{-1}(\text{coWP}(H, X) \cup A)$. This means that $\phi(\omega) = \tilde{\omega} \in \text{coWP}(H, X) \cup A$. If $\tilde{\omega} \in \text{coWP}(H, X)$, we have $\omega =_G \phi(\omega) = \tilde{\omega} \neq 1_H =_G 1_G$. Thus, $\omega \neq_G 1_G$.

If $\tilde{\omega} \in A$, we have $\omega =_G \phi(\omega) = \tilde{\omega} = \omega't$. Now, suppose $\omega =_G 1_G$. Then, we have $1_G = 1_H = \omega't$, implying $\omega' = t^{-1}$. But $\omega' \in H$ by construction, so that $t^{-1} \in H$ and therefore $t \in H$, meaning $t = 1$ (because $1, t \in T$), contradicting the fact that A is constructed in $T \setminus \{1\}$. Then, $\omega \neq_G 1_G$. \square

Note that we have proved in 4.1.6 the regularity of any set with all strings (finitely generated) and that any finite set $\{a_1, \dots, a_m\}$ is a regular language, as shown by the following finite state automaton.

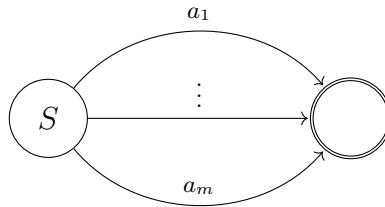


Figure 4.3 – Finite sets are regular languages.

4.4 Wreath Products

Definition 4.4.1. Given two groups G and H and a group homomorphism $\phi : H \rightarrow \text{Aut}(G)$, the *semidirect product* of G and H with respect to ϕ is the group $G \rtimes_{\phi} H =$

$(G \times H, \cdot)$, where

$$\begin{aligned} \cdot : (G \rtimes_{\phi} H) \times (G \rtimes_{\phi} H) &\rightarrow G \rtimes_{\phi} H \\ (g_1, h_1) \cdot (g_2, h_2) &\mapsto (g_1 \phi(h_1)(g_2), h_1 h_2) \end{aligned}$$

so that the identity element is $(1_G, 1_H)$ and the inverse of (g, h) is $(\phi(h^{-1})(g^{-1}), h^{-1})$.

Definition 4.4.2. Given two groups G and H , the *restricted standard wreath product* of G with H is the group

$$G \wr H = \left(\bigoplus_{h \in H} G \right) \rtimes_{\phi} H$$

where $\phi : H \rightarrow \text{Aut} \left(\bigoplus_{h \in H} G \right)$ take elements of H into permutations: $\phi(h)(t_l)_{l \in H} = (t_{hl})_{l \in H}$,

being the product of the indexes in H . The group $\left(\bigoplus_{h \in H} G \right)$ is called the *base* of the wreath product.

Theorem 4.4.3 (Holt-Rees-Röver-Thomas, [15]). *Take G a co-context-free group and H a context-free group. Then $G \wr H$ is a co-context-free group.*

Proof. Assume that G and H are generated, respectively, by S and Y . Then $G \wr H$ is generated by $S \cup Y$ and the base group is, by definition, the direct sum of copies G_h , $h \in H$, of G . We can understand the base group B as the set of all functions $b : H \rightarrow G$ such that $b(h) = 1_G$, except for finitely many $h \in H$. The group H acts on B , where the action of $h \in H$ on $b \in B$ is given by

$$h \circ b(h') = b(h'h^{-1}).$$

Such action defines the semidirect product which is equal to $G \wr H$. Let us identify G with the subgroup of B for which the elements such that $b(h)$ is trivial are all nontrivial $h \in H$. With this identification, any element $w \in G \wr H$ is of the form bh , for $b \in B$ and $h \in H$, being $w \neq 1_{G \wr H}$ if and only if $h \neq 1_H$ or $b(h') \neq 1_G$ for some $h' \in H$.

Before going on, let us consider $w = w_1 w_2 \cdots w_l \in (S \cup S^{-1} \cup Y \cup Y^{-1})^*$, where $w_i \in (S \cup S^{-1} \cup Y \cup Y^{-1})$ for any $i = 1, 2, \dots, l$. We define $w_{(i)}$ as the prefix of length i of w and \bar{w} as the word obtained from w by deleting all letters in the alphabet $S \cup S^{-1}$. As previously stated, $w \equiv_{G \wr H} bh$ for some $b \in B$ and $h \in H$. Thus, $\bar{w} \equiv_H h$. Fix some $h' \in H$ and let J be the subset $\{i_1, i_2, \dots, i_k\} = J$ of $\{1, 2, \dots, l\}$ such that $i_j < i_{j+1}$ for $1 \leq j < k$ and such that $\bar{w}_{(i)} \equiv_H h'^{-1}$, $w_i \in S \cup S^{-1}$ for any $i \in J$. Then $b(h') \equiv_G w_{i_1} w_{i_2} \cdots w_{i_k}$. This is the main fact that we use in the next steps of the proof.

Let us describe a nondeterministic pushdown automaton \mathcal{M} which accepts $\text{coWP}(G \wr H)$. First, being nondeterministic, it has two paths, one for checking if h is nontrivial, for a given $w \in G \wr H$, and the other to check if b maps some $h' \in H$ to a nontrivial element.

In order to test if h is trivial, all we need is to start at the bottom of the stack and then write on the stack any element from $Y \cup Y^{-1}$ we read in w , erasing it if it is the case that two consecutive elements on the stack would be the inverse of each other. Letters from w in $S \cup S^{-1}$ are ignored by doing nothing to the stack. If at the end of the input string the stack is not at the bottom symbol, then h was not trivial and w must be accepted. This behavior of the automaton is given by states q_0, q_1 and q_2 in Figure 4.5.

In order to test if there is some $h' \in H$ for which $b(h')$ is not trivial, our automaton must add nondeterministically a word $v \in (Y \cup Y^{-1})^*$ to the stack, on top of the bottom symbol $\$$ (see Figure 4.4), representing the element $h' \in H$.

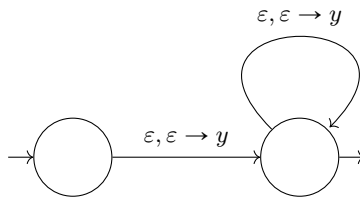


Figure 4.4 – **I** (one transition for each $y \in Y \cup Y^{-1}$)

After that, it starts reading the given w . Its behavior is:

- i. ignore any letter in $S \cup S^{-1}$ and read letters in $Y \cup Y^{-1}$, doing the same as it did on the other part (add the elements to the stack, unless they could be canceled, then erase the top of the stack element) until it reaches the bottom of the stack symbol.
- ii. Once the bottom symbol is seen, the automaton tests if the next letter is in $S \cup S^{-1}$. If so, the letter is written at the end of the input string of the pushdown automaton \mathcal{N} which accepts the language of the co-word-problem of G (remember that G is co-context-free by hypothesis) and the automaton tests the next letter. This process keeps being repeated until the first letter in $Y \cup Y^{-1}$ is read.
- iii. Once a letter in $Y \cup Y^{-1}$ is read, the automaton writes it on the stack and then goes back to step *i*.

The automaton keeps going on with this behavior until it finishes reading the input word. Once the end of the input string symbol $\#$ is reached (we add this symbol to the string alphabet before beginning), the automaton finishes by telling \mathcal{N} to start reading what is in its input string. \mathcal{N} then gives an answer of accepted if $b(h') \neq 1_G$, which our automaton returns to us as w nontrivial.

Notice that what this last part does is: step *i* tests for which $i \in \{1, 2, \dots, l\}$ the equality $\overline{w_{(i)}} \equiv_H h'^{-1}$ holds; step *ii* then writes the w_i with $i \in J$ in the input tape of \mathcal{N} ; and step *iii* merely works as intermediate step in between the finding of two different of

the desired i 's. Thus, what is written in the input tape of \mathcal{N} when it starts working is exactly $w_{i_1}w_{i_2}\cdots w_{i_k} \equiv_G b(h')$. The whole automaton is shown in Figure 4.5, together with Figure 4.4.

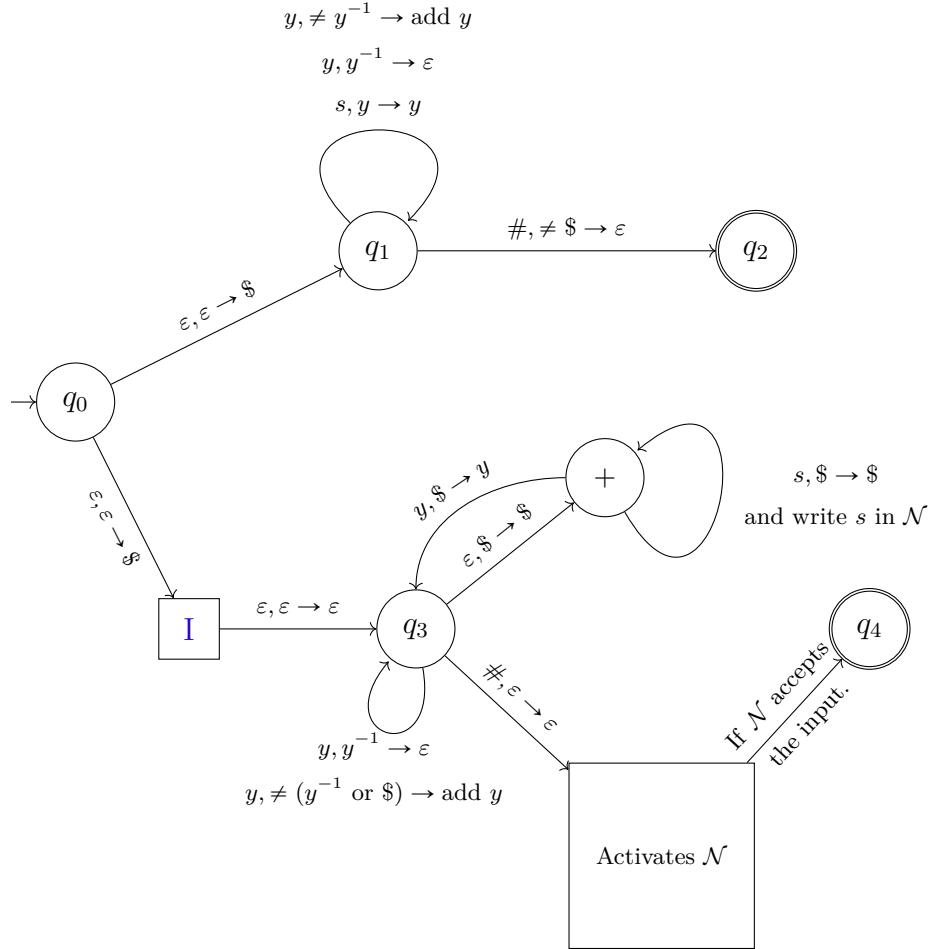


Figure 4.5 – \mathcal{M}

In the transitions, $b \rightarrow \text{add } c$ means that we add something to the top of the stack without erasing anything from it. Furthermore, the transitions hold for $y \in Y \cup Y^{-1}$ and $s \in S \cup S^{-1}$: we have written several transitions (one for each s and y) as only one, in the figure, for easier visualization. \square

4.5 Other Properties

In this section we make a compendium of interesting results we came across concerning the co-context-free groups. The proofs are left for the interested reader to go after, since they either are too technical to be of interest or require too much background that it is not within our purposes to introduce. The results may be found in [15].

Proposition 4.5.1. *Every co-context-free group has solvable word problem. Furthermore, the problem is solvable in cubic time with respect to the length of the input word.*

We also introduce two new problems and obtain results about them.

Question 4.5.2.

(a) Given H a subgroup of the group G , is it possible to decide whether, given $g \in G$, $g \in H$? (Generalised Word Problem)

(b) Given a finitely generated group G and $g \in G$, is it possible to decide if g has finite order? (Order Problem)

Proposition 4.5.3. *There exist co-context-free groups with unsolvable conjugacy and generalised word problems.*

Proposition 4.5.4. *Every co-context-free group has solvable order problem. Moreover, if the order is finite, then it can be determined.*

The next result is actually about one-counter languages, but it was put here because they are only a particular case of context-free languages.

Proposition 4.5.5. *Given G a finite group and H a virtually cyclic group, the group $G \wr H$ has one-counter co-word problem.*

We end this section, and this chapter, presenting groups which are not co-context-free groups, as well as some which are.

Theorem 4.5.6. *Given a finitely generated nilpotent group G , it is a co-context-free group if, and only if, it is virtually abelian.*

Corollary 4.5.7. *The Heisenberg group $H = \langle A, B, C \mid [A, B] = C, [A, C] = [B, C] = 1 \rangle$, where $[A, B] = A^{-1}B^{-1}AB$ is the commutator, is not a co-context-free group.*

Definition 4.5.8. A *Baumslag-Solitar group* is a group with presentation in the form $\langle a, b \mid b^{-1}a^mb = a^n \rangle$, where $m, n \in \mathbb{Z} \setminus \{0\}$.

Theorem 4.5.9. *A Baumslag-Solitar group is co-context-free if, and only if, it is virtually abelian.*

Definition 4.5.10. A group G is said to be *polycyclic* if it is solvable and every subgroup is finitely generated.

All polycyclic groups have finite presentations, so that it makes sense to talk about their word-problem language.

Theorem 4.5.11. *A polycyclic group has context-free co-word problem language if, and only if, it is virtually abelian.*

We finish this chapter stating some open questions about the co-context-free groups.

Question 4.5.12.

- (i) *Is the property of being co-context-free closed under taking free products? For example, is $\mathbb{Z} * \mathbb{Z}^2$ co-context-free? [15]*
- (ii) *Is Grigorchuk's group co-context-free? [5]*

Question (ii) is relevant in relation to Lehnert's conjecture, see Chapter 6, while question (i) deals with another possible closure property.

5. Thompson Groups

Thompson's groups are relevant in group theory because they are counter-examples to many conjectures, as they present many unusual properties. The most cited ones are the original ones: F, T and V , which are related to each other by containment, that is, $F \subset T \subset V$. In this chapter, we begin by introducing Thompson groups and a few known results about them. After that, we define one of their generalizations, the so called Higman-Thompson groups. The chapter ends with the introduction of the Houghton groups, which are subgroups of the Higman-Thompson groups. We introduce these groups in order to prove, in the next chapter, that they are all co-context-free.

5.1 Thompson Groups

The Group F

Here we present some properties of F , such as not containing non-abelian free subgroups. It is also known that F does not satisfy any nontrivial identity and that it has exponential growth. All these properties are briefly introduced only as interesting facts, but are not demonstrated, as our main goal is not related to them. The interested reader may consult [8] and [27], chapter 5. Let us begin by defining F . There are many equivalent definitions, being one or another more convenient according to what is needed to prove. However, we present only the definition which is of interest to us, the one that can be generalized in order to define T and V .

Definition 5.1.1. F is the group with all continuous increasing and piecewise linear functions $f : [0, 1] \rightarrow [0, 1]$ such that

- (i) f is a homeomorphism;
- (ii) numbers of the form $\frac{a}{2^b}$, $a, b \in \mathbb{N}$ are taken into numbers of the same form;
- (iii) the number of linear pieces is finite, each of them having slope 2^k , for some $k \in \mathbb{Z}$;
and
- (iv) the derivative is non-continuous only at points of the form $\frac{a}{2^b}$, $a, b \in \mathbb{N}$.

The operation is the composition of functions.

F can be said to be a 2-dimensional analogue for free groups. But the first property we present says that F is actually far from being free.

Theorem 5.1.2. *Every subgroup of F is either abelian or has subgroup isomorphic to $\mathbb{Z} \wr \mathbb{Z}$, thus being infinitely generated, abelian and free. Hence, F does not have non-abelian free subgroups.*

We finish the introduction of F by stating Abért's criterion, which can be used to obtain an interesting result.[27]

Definition 5.1.3. Let G be a group acting (on the right) on an infinite set X . We say that G *separates* X if for any subset $Y \subseteq X$, the set $G_Y = \{g \in G \mid y \cdot g = y \text{ for all } y \in Y\}$ does not fix any point outside Y , that is, for every $y \notin Y$, there exists $g \in G_Y$ such that $y \cdot g \neq y$.

Theorem 5.1.4 (Abért). *Given a group G and an infinite set X , if G separates X , then G does not satisfy any nontrivial group identity.*

The next result follows from Abért's criterion.

Theorem 5.1.5. *F does not satisfy nontrivial group identities.*

The Group T

The easiest way of defining T is through homeomorphisms of the unit circle that follow some rules. Let us so define it.

Definition 5.1.6. Let S^1 be the unit circle obtained from $I = [0, 1]$ by identifying the endpoints. Thompson's group T is the set of all piecewise linear homeomorphisms $f : S^1 \rightarrow S^1$ such that

- (i) f maps images in S^1 of numbers of the form $\frac{a}{2^b}$, $a, b \in \mathbb{N}$ to images of numbers of the same form;
- (ii) f is differentiable except at a finite number of points that are images of numbers of the form $\frac{a}{2^b}$, $a, b \in \mathbb{N}$; and
- (iii) in the differentiable intervals, the slope is 2^k , for some $k \in \mathbb{Z}$;

together with the group operation composition of functions.

It is possible to show that the following functions, A and B , generate a group isomorphic to F .

$$A(x) = \begin{cases} \frac{x}{2}, & 0 \leq x \leq \frac{1}{2} \\ x - \frac{1}{4}, & \frac{1}{2} \leq x \leq \frac{3}{4} \\ 2x - 1, & \frac{3}{4} \leq x \leq 1. \end{cases}$$

$$B(x) = \begin{cases} x, & 0 \leq x \leq \frac{1}{2} \\ \frac{x}{2} + \frac{1}{4}, & \frac{1}{2} \leq x \leq \frac{3}{4} \\ x - \frac{1}{8}, & \frac{3}{4} \leq x \leq \frac{7}{8} \\ 2x - 1, & \frac{7}{8} \leq x \leq 1. \end{cases}$$

One can induce elements \tilde{A} and \tilde{B} of T using A , B and the identification of S^1 with the unity interval $I = [0, 1]$. Then, defining the function

$$C(x) = \begin{cases} \frac{x}{2} + \frac{3}{4}, & 0 \leq x \leq \frac{1}{2} \\ 2x - 1, & \frac{1}{2} \leq x \leq \frac{3}{4} \\ x - \frac{1}{4}, & \frac{3}{4} \leq x \leq 1 \end{cases}$$

and inducing a function \tilde{C} in S^1 as well, we have that \tilde{A} , \tilde{B} and \tilde{C} generate T and \tilde{A} and \tilde{B} generate a subgroup of T which is isomorphic to F .

The Group V

Definition 5.1.7. Consider the unit circle S^1 . Thompson's group V is the set of all right-continuous bijective functions $f : S^1 \rightarrow S^1$ such that

- (i) f maps images of numbers of the form $\frac{a}{2^b}$, $a, b \in \mathbb{N}$ to images of numbers of the same form;
- (ii) f is differentiable except at a finite number of points which are images of numbers of the form $\frac{a}{2^b}$, $a, b \in \mathbb{N}$; and
- (iii) on every maximal interval of differentiability, the slope is 2^k , for some $k \in \mathbb{Z}$;

together with the group operation composition of functions.

We use the identification of S^1 as the quotient of $[0, 1]$ in order to define a new function π_0 .

$$\pi_0(x) = \begin{cases} \frac{x}{2} + \frac{1}{2}, & 0 \leq x < \frac{1}{2} \\ 2x - 1, & \frac{1}{2} \leq x < \frac{3}{4} \\ x, & \frac{3}{4} \leq x < 1. \end{cases}$$

It is then possible to prove that the previous functions \tilde{A} , \tilde{B} and \tilde{C} , together with the function π_0 (all seen as maps induced on S^1), generate V .

Before going on to the introduction of the Higman-Thompson groups, we give some more results and interesting facts about the Thompson groups. If nothing else is indicated, the results are in [8].

- The groups $[F, F]$, T and V are simple.
- F , T and V have exponential word growth.
- Every proper quotient of F is abelian.
- The groups F, T and V have solvable conjugacy problem. Several different proofs have now been found (see the reference list in [7] for more detailed references to such proofs).
- F has been thought as a potential counter-example for the *von Neumann-Day Conjecture*, which states that a group is non-amenable if and only if it contains a free group with two generators as subgroup. Nevertheless, the amenability of F is still an open question and the conjecture was disproved by Ol'shanskii through the Tarski monster groups, as one can find out in 5.8.5.2 from [27].

From here on, we will let F , T and V always denote the Thompson groups.

5.2 Higman-Thompson Groups

The Higman-Thompson Groups are defined as bijections on infinite sequences. They are of interest for us because V is one of them.

Definition 5.2.1. Take two fixed integers, $n \geq 2$ and $r \geq 1$, and two finite alphabets, $Q = \{q_1, \dots, q_r\}$ and $\Sigma = \{\sigma_1, \dots, \sigma_n\}$. Let $\Omega = Q\Sigma^{\mathbb{N}}$ be the set of infinite sequences starting with an element of Q followed by elements of Σ . A subset $B \subset Q\Sigma^*$ is called a

barrier if for any $\omega \in \Omega$, there is exactly one $b \in B$ with $\omega \in b\Sigma^{\mathbb{N}}$, that is, b is the unique prefix in B of ω .

Example 5.2.2. Barriers can easily be visualized using trees. In Figure 5.1, we have a tree picturing the set Ω for $Q = \{q_1, q_2\}$ and $\Sigma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$: each infinite path followed from the root of the tree is a sequence in Ω . In the figure, we omit the arrows coming out from some of the vertices for easier visualization, but it should be clear from the context where there are supposed to be arrows, since the tree is infinite.

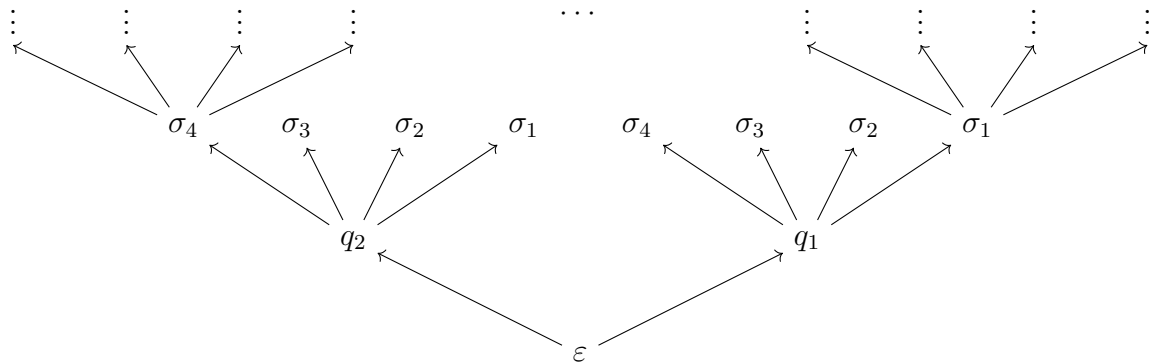
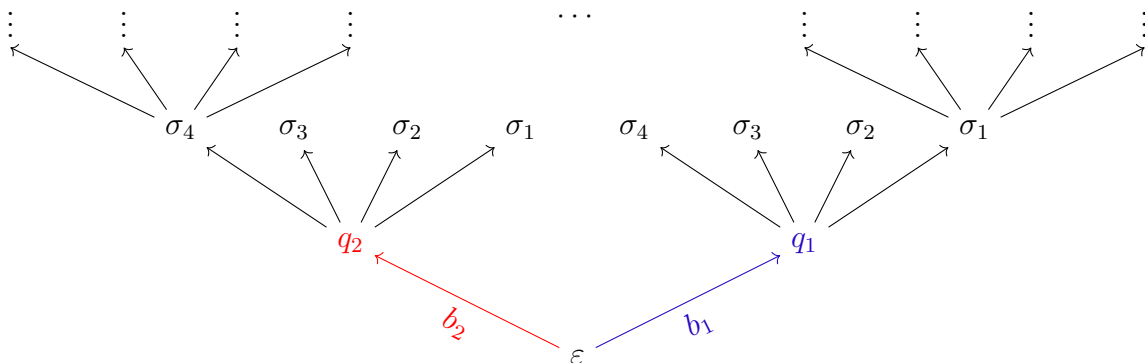


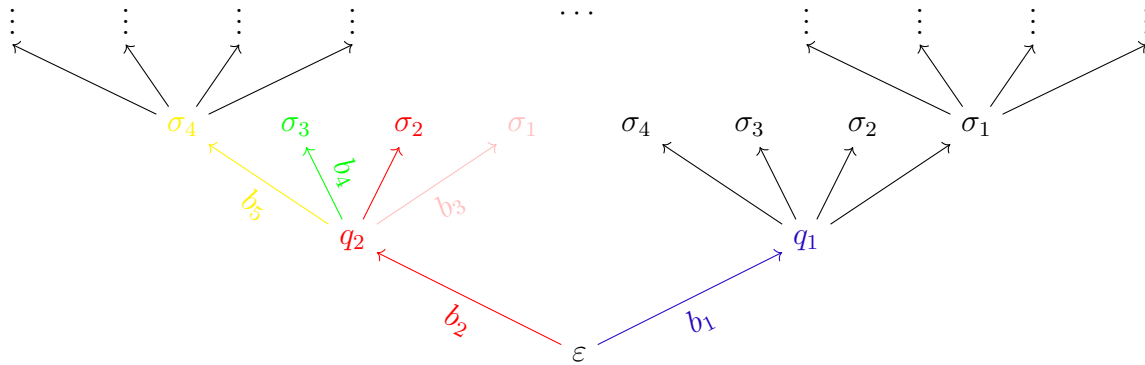
Figure 5.1 – Ω pictured as a tree.

In this case, a barrier is any finite subtree which has at least one path beginning with q_i , for each $q_i \in Q$. The uniqueness required in the definition is automatically given by the fact that the trees have no cycles and therefore it is not possible to build two distinct finite paths beginning in the root and finishing in the same vertex (if we do not chose paths finishing in a lower level of the tree where there is already a path finishing in a higher level). In Figure 5.2 we present two possible barriers for Ω as defined in this example.



(a) A barrier $B = \{b_1 = q_1, b_2 = q_2\}$.

Figure 5.2 – Examples of barriers for Ω .



(b) A barrier $B = \{b_1 = q_1, b_2 = q_2\sigma_2, b_3 = q_2\sigma_1, b_4 = q_2\sigma_3, b_5 = q_2\sigma_4\}$.

Figure 5.2 – Examples of barriers for Ω . (cont.)

Given two fixed integers and their two associated alphabets, Q and Σ , we denote by \mathcal{B}_c the set of all barriers of cardinality c .

Definition 5.2.3. Fix $n \geq 2$ and $r \geq 1$, integers. The *Higman-Thompson group* $G_{n,r}$ is the group $(G_{n,r}, \cdot)$ of functions defined as follows: given two barriers (for the same finite sets Q and Σ) of same cardinality $c < \infty$, B_1 and B_2 ; and a bijection $\phi : B_1 \rightarrow B_2$, there exists an induced bijection $g_\phi : \Omega \rightarrow \Omega$, which replaces the prefixes. $G_{n,r}$ is the the of all such induced bijections, that is

$$G_{n,r} = \bigcup_{c \in \mathbb{N}} \{g_\phi \mid \phi : B_1 \rightarrow B_2 \text{ is a bijection}\}_{B_1, B_2 \in \mathcal{B}_c},$$

with operation given by composition.

Some results about the Higman-Thompson groups:

- $G_{n,r}$ is finitely presented. [24]
- $G_{n,r}$ and $G_{n,s}$ are isomorphic if $r \equiv s \pmod{n-1}$. [13]
- Let $G_{n,r}^+$ be the commutator subgroup of $G_{n,r}$. Then $G_{n,r}^+$ and $G_{m,s}^+$ are isomorphic if, and only if, $m = n$ and $\gcd(n-1, r) = \gcd(n-1, s)$. [24]
- $G_{n,r}$ is simple if n is even and has a simple subgroup of index 2 if n is odd. [13]
- $G_{n,r}$ has solvable conjugacy and power conjugacy problems. [2].
- The group $G_{n,r}$ is the automorphism group of the free Cantor algebra $C_n[r]$ of type n on r generators. [13]
- V is $G_{2,1}$. [13]

The last result, about V being isomorphic to $G_{2,1}$, can be easily visualized using the action of V in the tree of standard dyadic intervals. First, remember from Example 5.2.2 that the barriers can be defined as subtrees of infinite trees in a natural way. Second, consider trees of the form in Figure 5.3. It is easy to notice, intuitively, that there must be a canonical bijection between the set of functions used to define the groups $G_{n,r}$ and the set of functions acting in these trees. This can in fact be shown and it is therefore possible to understand $G_{n,r}$ as a group acting in the tree in Figure 5.3.

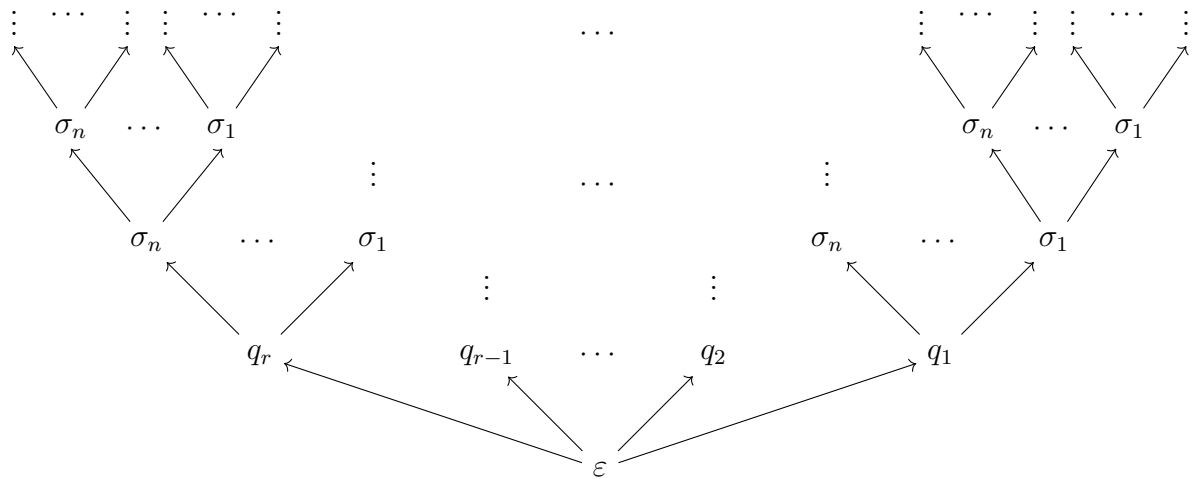


Figure 5.3 – A n, r tree.

We now recall that we gave the definition of functions \tilde{A} , \tilde{B} , \tilde{C} and π_0 (all seen as maps induced on S^1), which generate V , see subsection *The Group V*. There is also a canonical bijection between the standard dyadic partitions used to define these functions (and V) and an infinite binary tree, which is then called a *tree of standard dyadic intervals*, see [8] for more on that topic. Using this bijection, it is possible to picture the functions \tilde{A} , \tilde{B} , \tilde{C} and π_0 by their reduced tree diagrams, as in Figures 5.4, 5.5, 5.6 and 5.7.

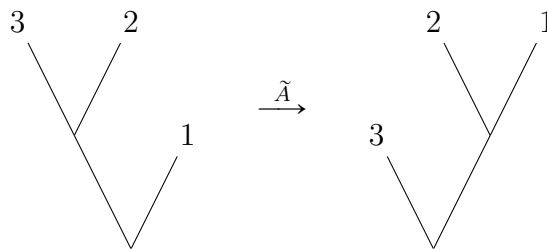
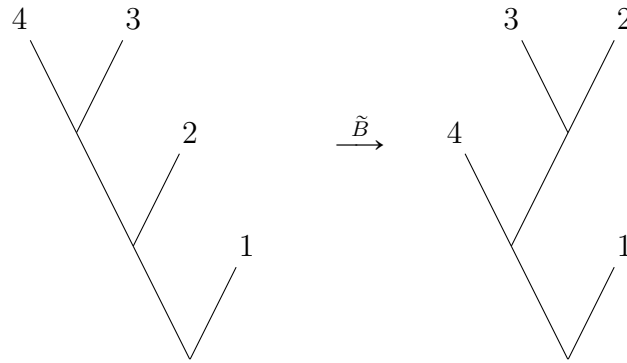
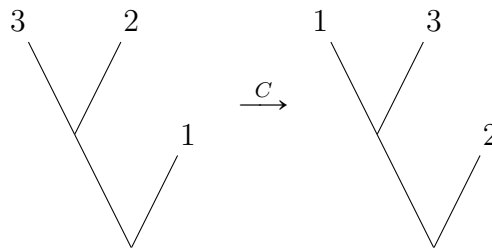
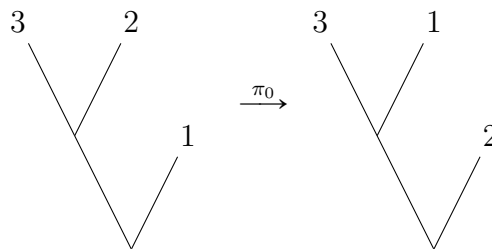


Figure 5.4 – Reduced tree diagram for \tilde{A} .

Figure 5.5 – Reduced tree diagram for \tilde{B} .Figure 5.6 – Reduced tree diagram for \tilde{C} .Figure 5.7 – Reduced tree diagram for π_0 .

Finally, it is easy to see that the action on a 2, 1 tree and the action on a binary tree (as the tree of standard dyadic intervals is) are equivalent: in the 2, 1 tree, use the only first level vertex q_1 as the root for the binary tree upon which we apply the action. These actions on trees picture the relation between $G_{2,1}$ and V for an intuitive understanding of their isomorphism.

5.3 Houghton Groups

Let us now define Houghton Groups, groups of quasi-automorphisms which shift points in a disjoint union of copies of an infinite graph.

Definition 5.3.1. Given $\Gamma = (V, E)$ a locally finite graph, a *quasi-automorphism* of Γ is a bijection $\phi : V \rightarrow V$ such that, except at finitely many vertices, if there is an edge $e \in E$ with $s(e) = u, t(e) = v$, then there is an edge $f \in E$ such that $s(f) = \phi(u), t(f) = \phi(v)$.

Thus, a quasi-isomorphism of a graph is a permutation of its vertices, preserving all but a finite number of adjacencies. The set of all quasi-automorphisms of a graph Γ is denoted $QAut(\Gamma)$.

We notice that, if Γ is finite, then $QAut(\Gamma) = \text{Sym}(V)$, being $\text{Sym}(V)$ the symmetric group which acts on V . Also, adding or removing a finite number of edges to Γ does not, up to isomorphism, alter $QAut(\Gamma)$.

Definition 5.3.2. Take \mathbb{N}_0 , the graph with vertices associated to non-negative integers and edges $e \in E$ such that $(s(e), t(e)) = (i, j)$ for all $i = j \pm 1$. We define the n -star, $*_0^n$ as the disjoint union of n copies of \mathbb{N}_0 . A point $(k, l) \in *_0^n$ is the vertex k of the l th copy of \mathbb{N}_0 .

Figure 5.8 shows a piece of $*_0^3$.

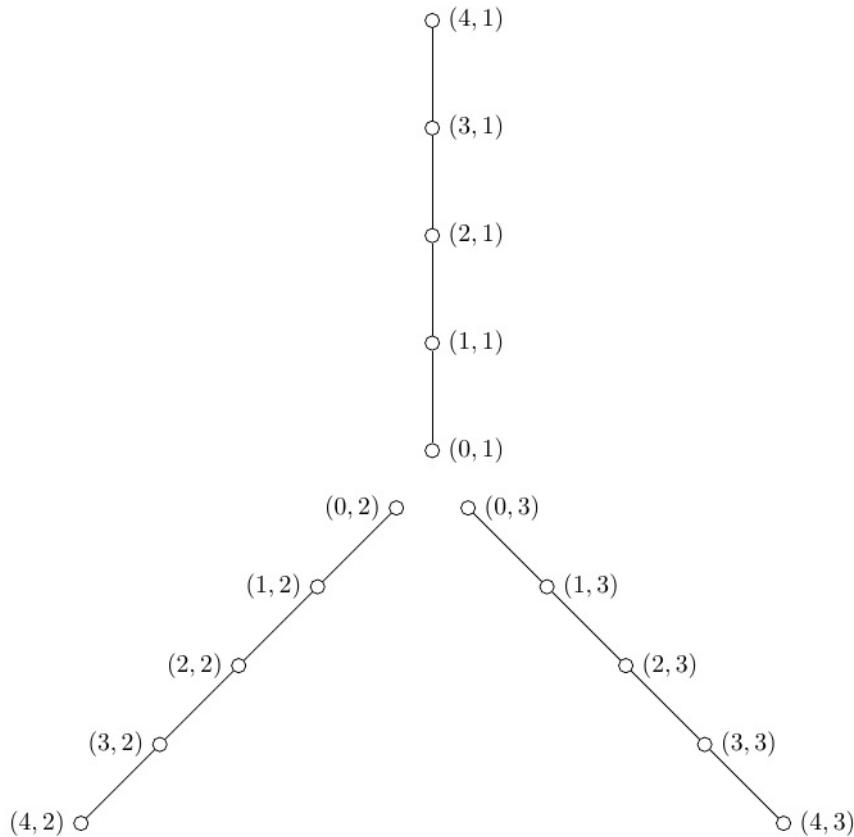


Figure 5.8 – $*_0^3$

Now, for each quasi-automorphism of $*_0^n$, one can choose a sufficiently large $r > 0$ and n integer numbers s_1, \dots, s_n such that, for some permutation of n elements, the effect of applying the function to a point greater than r is translating it by one of the chosen integers s_i and “rotating” it to another copy of \mathbb{N}_0 . The copy of \mathbb{N}_0 where the point will be taken to is the one corresponding to the image of the copy containing the original point

under the chosen permutation of n elements. That is, points are shifted along a semi-line and then entire semi-lines are permuted in the complement of a sufficiently large subgraph. Notice that the shifts along the semi-lines must balance each other, otherwise there will be points left in the subgraph of points lower than r which will be left without image. Putting it into symbols, we have: for each $\phi \in QAut(*_0^n)$, there exist $r > 0$, $s_1, \dots, s_n \in \mathbb{Z}$ and $\varphi \in S_n$ such that for all $k > r$, $\phi(k, l) = (k + s_l, \varphi(l))$ and $s_1 + \dots + s_n = 0$.

This can always be done because every quasi-automorphism is a permutation of the vertices in itself.

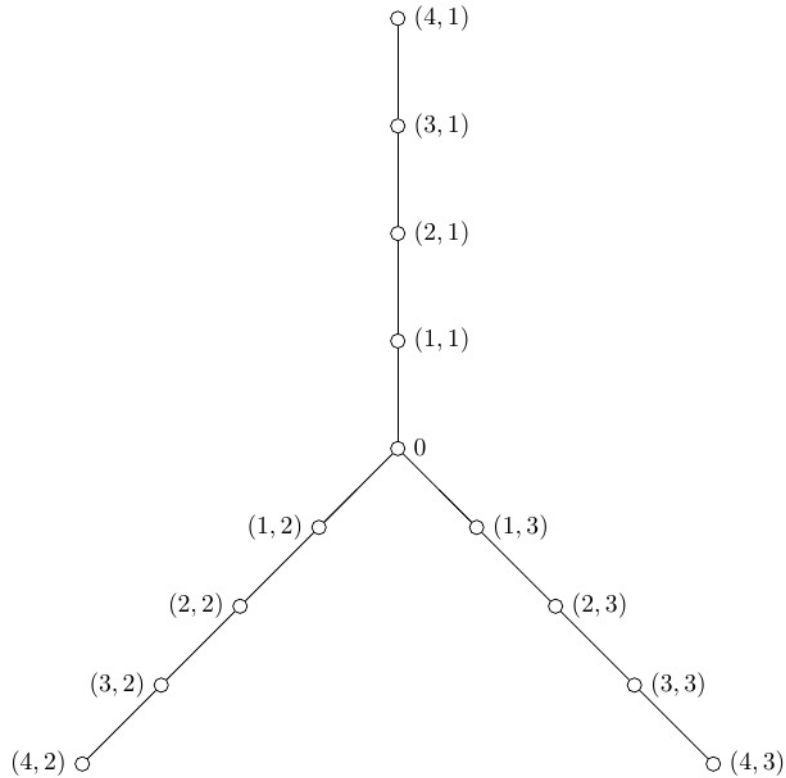
Definition 5.3.3. The n th Houghton group, H_n , is the group of all $\phi \in QAut(*_0^n)$ such that φ , as defined above, is the identity in S_n .

The Houghton groups were first introduced in [16], for the purpose of studying the first cohomology of groups with permutation module coefficients.

From the order of S_n , we get that H_n is a subgroup of finite index in $QAut(*_0^n)$, because any $\phi \in QAut(*_0^n)$ is obtained from composing the φ of the element from H_n that has the same integers s_1, \dots, s_n and the same $r > 0$ with some $\sigma \in S_n$. So, the number of cosets is equal to $|S_n| = n!$. Thus, from Theorem 4.3.2, we immediately obtain that H_n is co-context-free if, and only if, $QAut(*_0^n)$ is co-context-free.

Theorem 5.3.4 (Brown, [6]). *For $n > 1$, H_n is finitely generated.*

Better than just stating this fact, we can actually define the generators for H_n and they are exactly the shifts s_i , which we introduced above, for $n \geq 3$. We collapse all vertices with a 0 coordinate in $*_0^n$ to only one point, which we can do since n is finite and adding a finite number of edges, say between all the $(0, l)$, does not alter $QAut(*_0^n)$. Then, we obtain as result $*^n$, as exemplified in Figure 5.9. In $*^n$, the vertices are denoted by (k, l) for all $k \in \mathbb{N}$ and $l = 1, 2, \dots, n$ and we denote only by 0 the vertex where we collapsed the $(0, l)$ vertices of $*_0^n$.

Figure 5.9 – $*^3$

Notice that the shifts $s_i \in H_n$ are different for $*^n$ and for $*_0^n$. Let us work with the shifts s_i acting on $*^n$, as done in [18]. The action is, with the copies l modulo n :

$$s_i(k, l) = \begin{cases} (k, l) & \text{for } i \neq l \neq i + 1, \\ (k - 1, l) & \text{for } l = i, k \geq 2, \\ 0 & \text{for } l = i, k = 1, \\ (k + 1, l) & \text{for } l = i + 1, \end{cases}$$

and $s_i(0) = (1, i + 1)$. Then, for $n \geq 3$, the commutator $[s_i, s_{i+1}]$ acts as the transposition of 0 and $(1, i)$ and the group generated by all shifts contains all finite permutations, that is, is equal to H_n . For H_2 , we can consider $X = \{t, \tau\}$ as generating set, being $t = (i \mapsto i + 1)$ and $\tau = 0 \leftrightarrow 1$. Notice that t moves the vertices for the next one in a shift and τ can get out of a ray, going to zero, from where one could go into the other ray.

Röver showed in his PhD thesis that all Houghton groups embed into Thompson's group V (Proposition 2.6 in [26]).

Theorem 5.3.5 (Röver, [26]). *For $m \geq 1$, H_m is a subgroup of any $G_{n,r}$.*

6. Lehnert's Conjecture

In order to prove the main result studied in this work, that is, that the Thompson's group V is co-context-free, we need the following lemma.

Lemma 6.1. *Let L be a context-free language and L° be the set of all cyclic permutations of words in L , that is, $L^\circ = \{yx \mid x, y \in (S \cup S^{-1})^* \text{ and } xy \in L\}$, being S the alphabet for L . Then, L° is a context-free language.*

The proof is omitted, but can be found in [20].

Theorem 6.2 (Lehnert-Schweitzer, [18]). *The Higman-Thompson groups $G_{n,r}$ are co-context-free.*

Proof. It is proved in [13] that $G_{n,r}$ is finitely presented, so that we can consider a finite set of generators, X . We use the notations as presented in the definitions of Chapter 5, so that $\Sigma = \{\sigma_1, \dots, \sigma_n\}$ and $Q = \{q_1, \dots, q_r\}$ are finite sets and $\Omega = Q\Sigma^\mathbb{N}$. Our goal is to construct a context-free language L such that L° is $\text{coWP}(G_{n,r})$, in order to apply Lemma 6.1 and obtain the desired result.

Step 1 | Construction of the language L .

Take $\tau \in G_{n,r}$. By construction, we have that τ change the prefixes of every word $\omega \in \Omega$, by taking the prefix in the barrier B_1^τ into the prefix in the barrier B_2^τ . Notice that we added the index τ to the barriers, only to make it clear that we consider the two barriers which induce τ . There exists a constant $k_\tau \in \mathbb{N}$ such that for all $\omega \in \Omega$, the prefix $u \in B_1^\tau$ of ω is of length $lg(u) \leq k_\tau$, because the barrier is finite by definition.

Consider these constants k_τ for all the elements in the generating set and take their maximum, that is, take $k = \max_{x \in X \cup X^{-1}} k_x$. We then define the set of all infinite sequences with σ_1 in all entries of index greater than $k + 1$,

$$M = \{q\tilde{\omega}\sigma_1\sigma_1 \cdots \mid q \in Q, \tilde{\omega} \in \Sigma^k\}.$$

Notice that the cardinality of M is $|M| = rn^k$, because these are the only entries with possible variation, so that M is finite. We consider L as the set of words in $(X \cup X^{-1})^*$ which are not the identity on M , that is, for every element $g \in L$, there is at least one sequence in M that g does not fix. So,

$$L = \{\nu \in (X \cup X^{-1})^* \mid \nu|_M \neq Id|_M\}.$$

Step 2 | L is context-free.

Given $\omega \in M$, we show that the language L_ω of all words in $(X \cup X^{-1})^*$ which do not fix ω is context-free. After that, since $L = \bigcup_{\omega \in M} L_\omega$ and M is finite, we have that L is a finite union of context-free languages and, therefore, is context-free.

First notice that when we defined pushdown automata, the transitions were allowed to read at most one element of the stack and to write a word of any length. We could, instead of that, define transitions which read a word of length at most $k \in \mathbb{N}$ from the stack. This new definition allows the construction of a new type of automaton, but there is a correspondence between this new type of automata and pushdown automata, in the sense that for any context-free language, one can build automata of both types accepting the language. Moreover, for any automaton of the new type, there is a pushdown automaton accepting the same language. It is not difficult to prove that these automata are equivalent, we show in Figure 6.1 an example of how one transition of the automaton with access depth $k > 1$ is equivalent to a pushdown automaton with k transitions and viceversa.

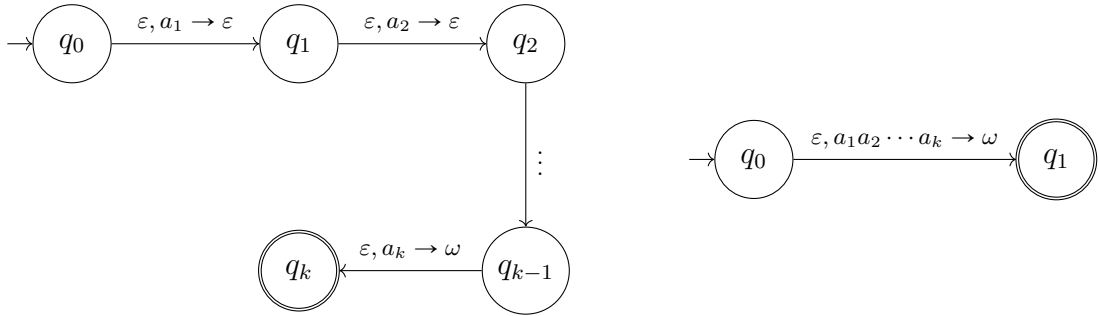


Figure 6.1 – Automata without and with access depth.

Thus, we can work with an automaton which has access depth k into the stack.

Our automaton then has tape alphabet equal to $X \cup X^{-1}$ and stack alphabet $\{q\} \cup \Sigma$, where $\omega = q\tilde{\omega}\sigma_1\sigma_1\cdots$. Given $\nu \in L_\omega \subset (X \cup X^{-1})^*$, ν is a word in the elements of the tape alphabet and therefore the input to the automaton is some word in $(X \cup X^{-1})^*$. We begin with $q\tilde{\omega}\sigma_1^l$ written in the stack, q in the top, for some fixed non-negative $l \in \mathbb{Z}$.

Let us describe how the automaton acts letter by letter when reading the input word. The transitions read $\tau \in X \cup X^{-1}$ from the tape and act on the stack by changing the prefix u into u' , according to the action of τ . Notice that, by construction of k , we are assured that τ has enough letters (to properly detect and change the prefix) in the stack if there are k or more symbols written there. If there are not enough symbols, let us say there are $\tilde{k} < k$ symbols forming the word a , the transition reads ε from the tape and change a into $a\sigma_1^{k-\tilde{k}}$. After that, the transitions go back to reading some τ from the tape and acting as the change of prefix. Thus, our transitions are of the form:

$$\begin{aligned} \tau, a \mapsto b, \quad \text{where } \tau \in X \cup X^{-1} \text{ and } \tau(a\sigma_1\sigma_1 \cdots) &= \tau(uv\sigma_1\sigma_1 \cdots) = \\ &= u'v\sigma_1\sigma_1 \cdots = b\sigma_1\sigma_1 \cdots \end{aligned}$$

and

$$\varepsilon, a \mapsto a\sigma_1\sigma_1 \cdots \sigma_1.$$

After reading the entire input word, the automaton checks what is on the stack. If the word written on the stack is $q\tilde{\omega}\sigma_1^m$ for some $m \in \mathbb{N} \cup \{0\}$, then the input word g acted as $g(\omega) = g(q\tilde{\omega}\sigma_1^l\sigma_1^\infty) = q\tilde{\omega}\sigma_1^m\sigma_1^\infty = \omega$ and it has fixed ω , being therefore not accepted. In any other case, the input word can be accepted.

Step 3 Finally, let us verify if $L^\circ = \text{coWP}(G_{n,r})$.

⊆ Consider $\lambda \in L^\circ$, $\lambda = vu$ such that $uv \in L$. By construction, we have that

$$uv(q\tilde{\omega}\sigma_1\sigma_1 \cdots) \neq q\tilde{\omega}\sigma_1\sigma_1 \cdots$$

for some $q\tilde{\omega}\sigma_1\sigma_1 \cdots \in M$. Consider the element $p = v(q\tilde{\omega}\sigma_1\sigma_1 \cdots) \in \Sigma$. Let us apply vu to p and verify that $vu(p) \neq p$:

$$vu(p) = vu(v(q\tilde{\omega}\sigma_1\sigma_1 \cdots)) = v(uv(q\tilde{\omega}\sigma_1\sigma_1 \cdots)) \neq v(q\tilde{\omega}\sigma_1\sigma_1 \cdots) = p.$$

Then, we have found p such that $vu(p) \neq p$ and therefore $vu = \lambda \in \text{coWP}(G_{n,r})$ and $L^\circ \subseteq \text{coWP}(G_{n,r})$.

⊇ First notice that $X \cup X^{-1} \subseteq L \cap L^\circ \cap \text{coWP}(G_{n,r})$ by construction of M , thus we need only to study the case $z \in \text{coWP}(G_{n,r}) \setminus X \cup X^{-1}$. Take $z \in \text{coWP}(G_{n,r}) \setminus X \cup X^{-1}$. We want to show that $z = yx$, where at least one of x and y is an element of length ≥ 2 and that there exists $\omega \in M$ such that $xy(\omega) \neq \omega$, implying $xy \in L$ and then $z \in L^\circ$. It suffices to show that there exists $\hat{\omega}$ with $yx(\hat{\omega}) \neq \hat{\omega}$ and $x(\hat{\omega}) = \omega \in M$, because then we have

$$\begin{aligned} \hat{\omega} \neq yx(\hat{\omega}) &= y(\omega) \Rightarrow \\ xy(\omega) &\neq x(\hat{\omega}) = \omega. \end{aligned}$$

For any $z \in \text{coWP}(G_{n,r})$, there is at least one non-fixed sequence $r_z = q\sigma_{i_1}\sigma_{i_2} \cdots$.

Take m_{min} as the minimal number such that no suffix of z induces a map sending the string $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_{m_{min}}}$ into a string of length less than k . As the sequence $q\sigma_{i_1}\sigma_{i_2} \cdots$ is infinite, it is always possible to take a sufficiently large prefix from it, in order to guarantee that the suffixes of z will map that suffix into a string of great enough length. Hence, the number m_{min} exists. Define $m = \max\{m_{min}, k\}$.

We want to ensure the existence of a suffix x of z or of z^{-1} which maps $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ to a string of length exactly k . After that, we conclude that ei-

ther z or z^{-1} is in L° and if necessary we use the fact that L° is closed under taking inverses to finish the proof.

Given $z \in \text{coWP}(G_{n,r})$, z is a sequence of elements in the generating set or its inverses. Each letter of z can have one of the following effects on r_z : change the prefix to a prefix of same length, change the prefix to a prefix of greater length or change the prefix to a prefix of smaller length. Considering the effect of all elements of z , we have a general behavior of z to shorten, increase or preserve the length of a prefix of r_z . In the case that z decreases the length of such prefix, we have that $m = m_{\min}$ and the length of $q\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_m}$ is decreased by z . Because $q\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_m}$ cannot be taken into a prefix of length $< k$ by construction, then we have that $m = m_{\min} > k$.

For the case where $m = m_{\min}$, without loss of generality, we write z for both z or z^{-1} , accordingly to the case. Let us see that the minimality of m ensure the existence of a suffix x of z which maps $q\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_m}$ to a string of length exactly k . Being m minimal, there is \bar{x} which takes $q\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_{m-1}}$ into a string of length $l < k$. Applying \bar{x} to $q\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_m}$ then results in a string of length $l + 1 \geq k$, because of the definition of m and because the addition of one more symbol at the end of the sequence does not change its prefix in the barrier, therefore it can not change the resulting image of \bar{x} except by the addition of another symbol if σ_{i_m} is part of the barrier prefix. But l and k are integers, thus $l < k$ together with $l + 1 \geq k$ implies $l + 1 = k$ and $x = \bar{x}$ is the desired suffix of z .

In the case that z increases the length of a prefix of r_z , consider z^{-1} instead of z . As z does not fix r_z , z^{-1} does not fix it either and because z has general behavior of increasing the prefix, z^{-1} have general behavior of decreasing the length. Thus, we can use for z^{-1} the same arguments which we use for a z which decreases the length. In this case, we prove that $z^{-1} \in L^\circ$ and the closure under taking inverses guarantee that $z \in L^\circ$.

Let us prove that L° is closed under taking inverses.

Consider $z^{-1} = a_n^{-1}a_{n-1}^{-1}\cdots a_2^{-1}a_1^{-1} \in L^\circ$. By definition of L° , at least one of the following holds:

$$\begin{aligned} z_1^{-1} &= a_1^{-1}(a_n^{-1}a_{n-1}^{-1}\cdots a_2^{-1}) \in L \\ z_2^{-1} &= a_2^{-1}a_1^{-1}(a_n^{-1}a_{n-1}^{-1}\cdots a_3^{-1}) \in L \\ z_3^{-1} &= a_3^{-1}a_2^{-1}a_1^{-1}(a_n^{-1}a_{n-1}^{-1}\cdots a_4^{-1}) \in L \\ &\vdots \\ z_{n-1}^{-1} &= (a_{n-1}^{-1}\cdots a_1^{-1})a_n^{-1} \in L. \end{aligned}$$

Being a language of elements which do not fix elements of M , L is closed under taking inverses, because the inverse of an element which does not fix some ω will not fix ω either. Thus, we have that for some $1 \leq i < n$, $z_i = a_{i+1}a_{i+2} \cdots a_n a_1 a_2 \cdots a_i \in L$. Then, $z = (a_1 a_2 \cdots a_i)(a_{i+1} a_{i+2} \cdots a_n) \in L^\circ$ and L° is closed under taking inverses.

Finally, in the case that z preserves the length of a prefix of r_z , we have $m = k$, because $m_{min} \leq k$ in that case, being the value of m_{min} equal to the least length that a prefix of r_z need to have in order for the action of the first element of z to be well defined. In this case, it is trivial that there exists a suffix of z which maps $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ to a string of length exactly k , because $x = z$ will always be such a suffix.

Now notice that all sequences with prefix $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ are mapped to a sequence with the same prefix as $z(r_z)$. That is true by the construction of k as the greatest possible length of prefixes in the barrier and by the definition of m . All the changes made by z in a sequence beginning with $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ change only this prefix and nothing on the subsequent elements of the sequence. And r_z is a non-fixed sequence, meaning that the action of z in $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ is not the identity and so all the sequences starting with $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ are non-fixed sequences.

Take $\hat{\omega} = q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}\sigma_1\sigma_1\sigma_1 \cdots$. The prefix $q\sigma_{i_1}\sigma_{i_2} \cdots \sigma_{i_m}$ is taken by x to a prefix of length exactly k by choice of x , thus $x(\hat{\omega}) = \omega \in M$. And it is a non-fixed sequence, so that $z(\hat{\omega}) = yx(\hat{\omega}) \neq \hat{\omega}$. Thus, $xy \in L$ and $z \in L^\circ$, meaning $\text{coWP}(G_{n,r}) \subseteq L^\circ$.

Therefore, $\text{coWP}(G_{n,r}) = L^\circ$ and we have finished the proof. \square

Remark 6.3. We use left actions. This means a word is read from right to left when acting as a function and from left to right when it is just a sequence. If one were to use right actions, the proof would not be changed in essence, but attention should be paid to the fact that the suffixes would become prefixes.

As an immediate corollary of Theorem 6.2 we get the main result of this chapter.

Corollary 6.4 (Lehnert-Schweitzer, [18]). *Thompson's group V is co-context-free.*

The result follows since the group $G_{2,1}$ is Thompson's group V . During the studies leading to that theorem, Lehnert formulated a statement that is equivalent to the following conjecture, which is nowadays the main open question in the construction of the Chomsky Hierarchy for groups. The statement was first presented in his PhD thesis, [17].

Conjecture 6.5 (Lehnert, [17]). *Let G be a finitely generated group. G is co-context-free if, and only if, it is a subgroup of Thompson's group V .*

This is not the original statement of the conjecture. Let $\mathcal{T}_{2,c}$ be the infinite rooted 2-edge-coloured binary tree, that is, an infinite rooted binary tree where we paint the vertices originated from left edges with colour 1 and the vertices originated from right edges with another colour 2. The original conjecture asked if all co-context-free groups were finitely generated subgroups of the group $QAut(\mathcal{T}_{2,c})$ of all bijections on the vertices of $\mathcal{T}_{2,c}$ which respect the edge and colour relationships, except for at possibly finitely many locations. It was proved in [5] that such group embeds in V and viceversa, so that the conjecture as stated here is equivalent.

We notice that a positive answer to (ii) of Question 4.5.12 would imply that Lehnert's conjecture is false, because Grigorchuk's group cannot be embedded in Thompson's group V .

By combining together Theorems 5.3.4, 5.3.5 and 4.1.6, we immediately see that for $n \geq 2$ every Houghton group H_n is co-context-free.

To conclude this chapter, we provide a direct proof of this statement as it is another illustration of the techniques developed in Theorem 6.2.

Corollary 6.6 (Lehnert-Schweitzer, [18]). *Let $n \geq 2$. Then the Houghton group H_n is co-context-free.*

Proof. Consider $S = \{s_1, s_1^{-1}, s_2, s_2^{-1}, \dots, s_n, s_n^{-1}\}$, being s_i the generators of H_n , as defined in Chapter 5, for $n \geq 3$. Let $L = \{\omega \in S^* \mid \omega(0) \neq 0\}$ be the set of words for which 0 is not a fixed point. We prove that L is context-free and that L° is the co-word problem for H_n , so that Lemma 6.1 ends the proof.

Step 1 | L is context-free.

We in fact show that L is a one-counter language, with stack alphabet $\Sigma = \{\$, 1\}$. In order to make the notation clear, let us denote the word

$$\underbrace{11 \cdots 1}_{m \text{ times}} \in \Sigma^*$$

as $m \in \Sigma^*$. Then, we can denote a word $\omega \in \{1\}^*$ which is shorter by 3 terms than m by $m - 3$ and so on, meaning that we consider a bijection from $\{1\}^*$ to \mathbb{N} . (Analogously, one could instead define the transition relations to be

$$\delta \subseteq ((Q \cup \{\varepsilon\}) \times (X \cup \{\varepsilon\}) \times (\Sigma^*)) \times (Q \times \Sigma^*),$$

while in the original definition of pushdown automata they were

$$\delta \subseteq ((Q \cup \{\varepsilon\}) \times (X \cup \{\varepsilon\}) \times (\Sigma \cup \{\varepsilon\})) \times (Q \times \Sigma^*),$$

with the third element being now possibly a whole word instead of just a letter. See Figure 6.1 for more on this equivalence.)

Now, the inverse functions of the shifts s_i are given by

$$s_i^{-1}(k, l) = \begin{cases} (k, l) & \text{for } i \neq l \neq i + 1, \\ (k + 1, l) & \text{for } l = i, \\ 0 & \text{for } l = i + 1, k = 1, \\ (k - 1, l) & \text{for } l = i + 1, k \geq 2 \end{cases}$$

and $s_1^{-1}(0) = (1, i)$. Then, we construct the one-counter (pushdown) automaton

$$\mathcal{M} = (Q, X, \Sigma, \delta, q_0, \$, F)$$

where

1. $Q = \{0, 1, 2, \dots, n, \star\}$, 0 being the initial state;
2. $X = S$;
3. $\Sigma = \{1, \$\}$ is the stack alphabet, $\$$ being the bottom symbol;
4. δ is as in the Figure 6.2;
5. $F = \{\star\}$ is the set of accepted states.

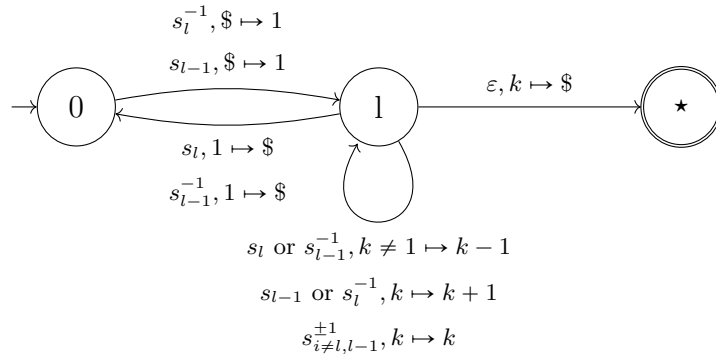


Figure 6.2 – \mathcal{M}

Thus, the given automaton receives words $\omega \in S^*$ and apply them to 0, accepting them only if they end in some vertex other than 0. That means the automaton accepts exactly L and so L is context-free.

Step 2 $L^\circ = \text{coWP}(H_n)$.

We will show the equality without paying attention to the generating set because we have already proved that being context-free is a property independent of generating set, so we can carry out this proof using S .

\subseteq Take a word $yx \in L^\circ$, implying $\omega = xy \in L$. Then, ω is not the identity (otherwise, $\omega(0) = 0$) and is therefore in the set $\text{coWP}(H_n, S)$. We have:

$$xy \neq Id \Rightarrow y \neq x^{-1} \Rightarrow yx \neq x^{-1}x \Rightarrow yx \neq Id.$$

Thus, $yx \in \text{coWP}(H_n, S)$.

\supseteq Consider $\omega \in \text{coWP}(H_n, S)$. Then, there is at least one point $p \in *^n$ such that $\omega(p) \neq p$. We show that there exists a suffix x such that $\omega = yx$ and $x(p) = 0$. With this result, we have that

$$\left. \begin{array}{l} x(p) = 0 \\ xy(0) = xy(x(p)) = x(yx(p)) \\ yx(p) \neq p \end{array} \right\} \Rightarrow xy(0) \neq 0.$$

Hence $xy \in L$ and $yx = \omega \in L^\circ$. It remains to be proved that such a suffix x does exist.

If the non-fixed point is $q = (q_1, q_2)$, we have two cases. Case 1: there exists x a suffix for ω such that $x(q) = 0$ and we have finished. Case 2: there is no such a suffix x for ω with $x(q) = 0$. In this case, we remember that there is r large enough with $\omega(k, q_2)$ acting as a shift for $k > r_{q_2}$, from the construction of H_n . Thus, there also exists a ray (\cdot, z_j) on which ω acts as an outbound shift for (k, z_j) , $k > r_{z_j}$, for some $1 \leq j \leq n$. This means that there is at least one point $p = (p_1, p_2)$, $p_2 \neq z_j$, whose image is on the ray (\cdot, z_j) , because our function ought to be a bijection. Thus, there exists some suffix x of ω with $x(p) = 0$, as desired.

Now it remains to be seen that H_2 is co-context-free. For this, we make use of the fact that $H_2 \cong S_\infty \rtimes \mathbb{Z}$, where S_∞ is the symmetric group of \mathbb{N} , so that each element $g \in H_2$ can be written as (σ, s) , $\sigma \in S_\infty$ and $s \in \mathbb{Z}$. Thus, for g to be nontrivial, we need it to have $s \neq 0$ or σ non trivial. Let us build a pushdown automaton $\widetilde{\mathcal{M}}$ which recognizes $\text{coWP}(H_2)$.

For that, we need a new symbol $\#$ meaning the end of the input string and a somewhat different notation for the transitions: we are able to add a symbol to the top of the stack without erasing the previous symbol by writing $b \rightarrow \text{add } c$. Our automaton reads g as a string in the alphabet of generators $X = \{t, \tau\}$. We use $H_2 \cong S_\infty \rtimes \mathbb{Z}$ only to understand how the automaton should work. Thus, to test if $s = 0$, the automaton needs to check what the sum of the exponents of t in g is. It is 0 if and only if $s = 0$. If $s \neq 0$ it is clear that g is nontrivial and should be accepted.

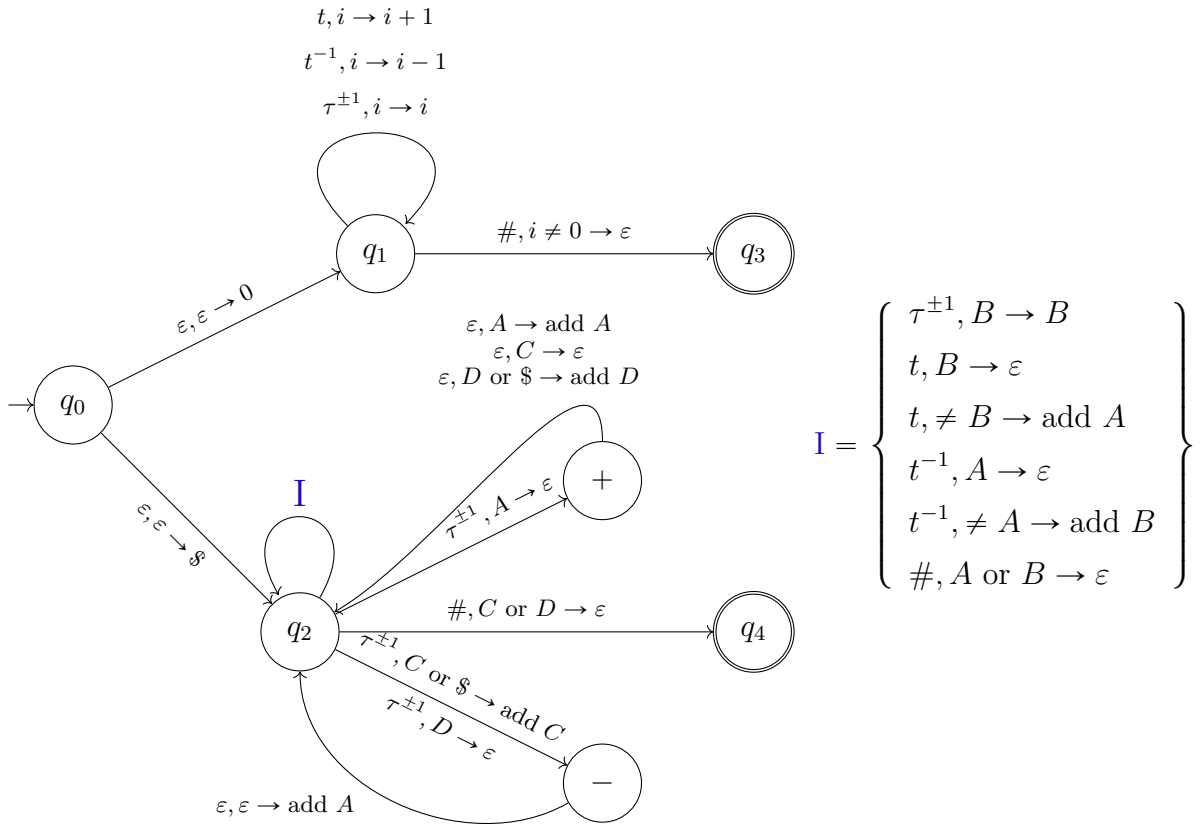


Figure 6.3 – $\tilde{\mathcal{M}}$

Notice that the automaton in Figure 6.3 is a nondeterministic one, with q_0 as starting state. The stack alphabet is $\Sigma = \{A, B, C, D, 1, \$\}$, where we can interpret A as $+1$, B as -1 , C as $0 \mapsto 1$, D as $1 \mapsto 0$ and we use an analogous bijection to the one between $\{1\}^*$ and \mathbb{N} used in *step 1* (now the bijection is with \mathbb{Z}).

The piece of the automaton that is composed by the states q_0, q_1 and q_3 is accountable for the verification if $s \neq 0$ or not, accepting the word as nontrivial immediately if it is the case.

The piece composed by the states $q_0, q_2, +, -$ and q_4 checks if σ is nontrivial, regardless of what is happening with s . The idea is that the automaton remembers the number of $\tau^{\pm 1}$ which have effect by adding C for right moves or D for left moves only when the stack contains at most one A and no B . C 's and D 's will cancel each other (as A 's and B 's will) and if there is any C or D left, that means σ was not trivial. The A 's and B 's are there as consequence of reading t and t^{-1} , respectively, and the $+$ and $-$ states are just auxiliary in order to act with depth in the stack. \square

Final Remarks

In this master thesis, we present the construction of Chomsky's Hierarchy for groups. We develop it step by step, presenting briefly the historical background of each result. We give closure properties on each type of groups or languages that we construct. We end by constructing the most recent result in the field and stating the main open question regarding the hierarchy. Let us recall the topics presented in the chapters of this dissertation.

In Chapter 2, we present the first result hinting at a possible connection between Chomsky's Hierarchy for languages and an analogous classification of groups, Anisimov's theorem. In Chapter 3, we introduce the second step in the classification of groups through their language of the Word Problem, the result by Muller & Schupp. We also explore the context-free languages and their properties, from both an automata and a grammar point of view.

We have also studied the proofs leading to Muller & Schupp's result, but unfortunately, due to time constraints, it was not possible to include all proofs in the present thesis, since they would need a whole new body of theory to be clearly presented.

In Chapter 4, we define the co-context-free groups and prove the main known closure properties, such as closure under taking direct products and wreath products. In chapter 5 we present definitions and properties of Thompson groups F , T and V , as well as those of Houghton groups and of Higman-Thompson groups.

Finally, in Chapter 6, we present the proof that Thompson's group V is co-context-free, which hints at the motivation for Lehnert's conjecture about the complete classification of the co-context-free groups.

As a perspective for the future, this work may lead to a PhD thesis exploring possible counter-examples for Lehnert's conjecture (or providing new interesting examples of co-context-free groups).

Bibliography

- [1] ANĪSĪMOV, A. V. The group languages. *Kibernetika (Kiev)*, 4 (1971), 18–24.
- [2] BARKER, N., DUNCAN, A. J., AND ROBERTSON, D. M. The power conjugacy problem in Higman-Thompson groups. *Internat. J. Algebra Comput.* 26, 2 (2016), 309–374.
- [3] BERSTEL, J. *Transductions and context-free languages*, vol. 38 of *Leitfäden der Angewandten Mathematik und Mechanik [Guides to Applied Mathematics and Mechanics]*. B. G. Teubner, Stuttgart, 1979.
- [4] BLEAK, C. Topics in groups course notes. , 2016. [Online; accessed 19-December-2018].
- [5] BLEAK, C., MATUCCI, F., AND NEUNHÖFFER, M. Embeddings into Thompson’s group V and coCF groups. *J. Lond. Math. Soc. (2)* 94, 2 (2016), 583–597.
- [6] BROWN, K. S. Finiteness properties of groups. In *Proceedings of the Northwestern conference on cohomology of groups (Evanston, Ill., 1985)* (1987), vol. 44, pp. 45–75.
- [7] BURILLO, J., MATUCCI, F., AND VENTURA, E. The conjugacy problem in extensions of Thompson’s group F . *Israel J. Math.* 216, 1 (2016), 15–59.
- [8] CANNON, J. W., FLOYD, W. J., AND PARRY, W. R. Introductory notes on Richard Thompson’s groups. *Enseign. Math. (2)* 42, 3-4 (1996), 215–256.
- [9] CHOMSKY, N. Three models for the description of language. *IRE Transactions on information theory* 2, 3 (1956), 113–124.
- [10] DIEKERT, V., AND WEISS, A. Context-free groups and their structure trees. *Internat. J. Algebra Comput.* 23, 3 (2013), 611–642.
- [11] GINSBURG, S. *The mathematical theory of context-free languages*. McGraw-Hill Book Co., New York-London-Sydney, 1966.
- [12] HERBST, T. On a subclass of context-free groups. *RAIRO Inform. Théor. Appl.* 25, 3 (1991), 255–272.
- [13] HIGMAN, G. *Finitely presented infinite simple groups*. Department of Pure Mathematics, Department of Mathematics, I.A.S. Australian National University, Canberra, 1974. Notes on Pure Mathematics, No. 8 (1974).

-
- [14] HOLT, D. F., REES, S., AND RÖVER, C. E. *Groups, languages and automata*, vol. 88 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2017.
- [15] HOLT, D. F., REES, S., RÖVER, C. E., AND THOMAS, R. M. Groups with context-free co-word problem. *J. London Math. Soc. (2)* 71, 3 (2005), 643–657.
- [16] HOUGHTON, C. The first cohomology of a group with permutation module coefficients. *Archiv der Mathematik* 31, 1 (1978), 254–258.
- [17] LEHNERT, J. *Gruppen von quasi-Automorphismen*. PhD thesis, Johann Wolfgang Goethe-Universität, 2008.
- [18] LEHNERT, J., AND SCHWEITZER, P. The co-word problem for the higman-thompson group is context-free. *Bull. London Math. Soc.* 39, 2 (2007), 235–241.
- [19] LÖH, C. *Geometric group theory*. Universitext. Springer, Cham, 2017. An introduction.
- [20] MASLOV, A. N. The cyclic shift of languages. (Russian). *Problemy Peredači Informacii* 9, 4 (1973), 81–87.
- [21] MEIER, J. *Groups, graphs and trees*, vol. 73 of *London Mathematical Society Student Texts*. Cambridge University Press, Cambridge, 2008. An introduction to the geometry of infinite groups.
- [22] MULLER, D. E., AND SCHUPP, P. E. The theory of ends, pushdown automata, and second-order logic. *Theoret. Comput. Sci.* 37, 1 (1985), 51–75.
- [23] MULLER, D. E., AND SCHUPP, P. E. A. Groups, the theory of ends, and context-free languages. *J. Comput. System Sci.* 26, 3 (1983), 295–310.
- [24] PARDO, E. The isomorphism problem for Higman-Thompson groups. *J. Algebra* 344 (2011), 172–183.
- [25] ROBINSON, D. J. S. *A course in the theory of groups*, second ed., vol. 80 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1996.
- [26] RÖVER, C. H. *Subgroups of Finitely Presented Simple Groups*. PhD thesis, Pembroke College, University of Oxford, 1999.
- [27] SAPIR, M. V. *Combinatorial algebra: syntax and semantics*. Springer Monographs in Mathematics. Springer, Cham, 2014. With contributions by Victor S. Guba and Mikhail V. Volkov.
- [28] SIPSER, M. *Introduction to the Theory of Computation*, 3rd ed. Cengage Learning, 2013.