UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de Formació Interdisciplinària Superior

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO
DEPARTAMENTO DE MATEMÁTICA

Bachelor's degree thesis

Malnormal closure computation

Efficient algorithm to compute the malnormal closure for finitely generated subgroups of finitary free products of finite and cyclic groups

Cesc Folch Aldehuelo

Advised by:

Pedro Ventura Alves da Silva (FCUP)
Enric Ventura Capell (UPC)

In partial fulfillment of the requirements for the
Bachelor's degree in Mathematics
Bachelor's degree in Informatics Engineering

May 2020

Abstract

In this thesis we take an existing polynomial algorithm to compute the malnormal closure of a finitely generated subgroup of a free group and improve its time complexity. In the last section we describe a graphical way to represent finitely generated subgroups of finitary free product of finite groups and cyclic groups, a family of groups containing the free groups. Using this graphical representation we can compute the malnormal closure of these subgroups with a similar algorithm to the one for the free groups, with the same time complexity.

keywords: *malnormal subgroup, malnormal closure, free group, free product, finite group, cyclic group, algorithm, Stallings graph*

MSC2020: 20-08

Acknowledgements

First of all, I would like to thank Prof. Pedro V. Silva for all the time, patience and advices he gave me during my stay in Porto, and most of all, for all the help during the righting of this thesis.

I would also like to acknowledge Prof. Enric Ventura, who first introduced me to free groups and Stallings graphs, and who has followed my progress during this year.

Special thanks to my flatmates here in Porto, they have been my family and friends during all these months in Portugal. Thank you for your company and patience, especially during these lasts months of quarantine.

Finally, I would like to thank my family, they have always been there to support and encourage me in all I decided to do. Thank you for all your love and for listening to me when I talk about math or computer science, even when you are not understanding a word of what I am saying.

Contents

1 Introduction

In [9] Stallings presented a geometrical approach to study finitely generated subgroups of free groups. Some years later, Kapovich and Myasnikov presented a combinatorial view of this concept and related some properties of the so-called Stallings graphs with properties of the subgroups they were representing. This was done in [4]. In this same paper they proved that malnormality and the malnormal closure computation where decidable problems for subgroups of the free group. In [8], Silva and Weil presented a polynomial algorithm to compute the malnormal closure for subgroups of the free group.

This thesis focus is the improvement of the time complexity of the algorithm for malnormal closure computation presented by Silva and Weil. After that, in the last section, we extend this algorithm to finitely generated subgroups of finitary free product of finite groups and cyclic groups. To do so, we will develop a geometric representation for these subgroups and then extend our algorithm to work with this new geometric representation.

The work is divided in 5 sections, the first 3 are introductory. The work will be divided as follows:

- Malnormality. In this section we will define the concept of malnormal subgroup and malnormal closure.
- Labeled graphs. For the geometric representation of subgroups we will be using labeled graphs, we will define them and prove some useful properties.
- Data structures and methods. To improve the algorithm time complexity we will need some well known data structures. In this section we will do a brief presentation for each of these data structures.
- Free group. The first part of this section is the presentation of Stallings graph. After that we relate them to malnormality and present the first algorithm from Silva and Weil. After that, we develop some methods and use them to improve the time complexity.
- Finitary free product of cyclic and finite groups. As the previous one, the first part of the section is focused in the development of a geometrical representation for finitely generated subgroups. After that we relate the geometrical view of subgroups to malnromality and show a first algorithm. The last part is an improvement of the time complexity of this algorithm using the same new methods as for the free group.

During the first 3 sections and the first half of the fourth section, until the presentation of Silva's and Weil's algorithm, we will be presenting previous concepts and algorithms that will be useful for our new work. The last part of the fourth and the fifth section are part of the new work developed in this thesis.

2 Malnormality

In this section we define the basic concepts of malnormality and some useful lemmas that we will use in further sections.

Definition 2.1 (Malnormal subgroup)**.** *Let $H \leq G$, we call this subgroup malnormal if for all $g \in G \setminus H$ we have $gHg^{-1} \cap H = 1$.*

Fact 2.2. *G is a malnormal subgroup of G.*

Fact 2.3. *Let $H, K \leq G$ be two malnormal subgroups, then $H \cap K$ is a malnormal subgroup.*

Now we can define the malnormal closure of a subgroup.

Definition 2.4 (Malnormal closure)**.** *Let $H \leq G$ and let H_{mal} be defined as follows:*

$$H_{mal} = \cap\{K \mid H \leq K \leq G \text{ with } K \text{ malnormal in } G\}$$

We call H_{mal} the malnormal closure of H.

Lemma 2.5. *If $g \in G \setminus H$ satisfies $gHg^{-1} \cap H \neq 1$ then $g \in H_{mal}$.*

Proof. Suppose that $g \notin H_{mal}$, then $g \in G \setminus H_{mal}$. But $1 \neq gHg^{-1} \cap H \leq gH_{mal}g^{-1} \cap H_{mal} = 1$ leads to a contradiction, so $g \in H_{mal}$. □

Corollary 2.6. *If $g \in G \setminus H$ satisfies $gHg^{-1} \cap H \neq 1$ then $H'_{mal} = H_{mal}$, where $H' = \langle H, g \rangle$.*

Proof. From Lemma 2.5 we have $H \leq H' \leq H_{mal}$, with this and the Definition 2.4 we get $H'_{mal} = H_{mal}$. □

3 Labeled graphs

In this section we are going to define the different types of graphs we are going to use to represent subgroups of the different types of subgroups. In each section we will define new operations on graphs to work with the subgroups of that family of groups.

Definition 3.1 (Graph)**.** *A graph Γ consists of two sets $E(\Gamma)$ and $V(\Gamma)$, and two functions $E(\Gamma) \to E(\Gamma)$ and $E(\Gamma) \to V(\Gamma)$: for each $e \in E(\Gamma)$ we have an element $\overline{e} \in E(\Gamma)$ and an element $\iota(e) \in V(\Gamma)$, such that $\overline{\overline{e}} = e$ and $\overline{e} \neq e$. The elements of $E(\Gamma)$ are called edges, and an $e \in E(\Gamma)$ is a directed edge of Γ, $\overline{e}$ is the reverse (inverse) edge of e. The elements of $V(\Gamma)$ are called vertices, $\iota(e)$ is the initial vertex of e, and $\tau(e) = \iota(\overline{e})$ is the terminal vertex of e. We call them the endpoints of the edge e. We call the degree of a vertex $v \in V(\Gamma)$ the number of different edges $e \in E(\Gamma)$ with $\iota(e) = v$.*

We will represent the graphs graphically using circles with an identifier inside to represent vertices and arrows to represent edges. An arrow that goes from point x and is pointing to point y represents a directed edge $e \in E(\Gamma)$ such that $\iota(e) = x$ and $\tau(e) = y$.

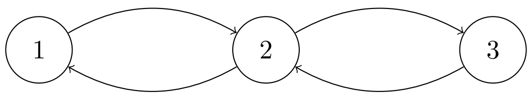


Figure 1: A graph with three vertices and four edges, each edge has an inverse

Definition 3.2 (Subgraph)**.** *A subgraph of Γ is a graph C such that $V(C) \subseteq V(\Gamma)$ and $E(C) \subseteq E(\Gamma)$. In this case, we write $C \subseteq \Gamma$.*

Given a finite set $X = \{a_1, a_2, ..., a_n\}$ we define the set X^{-1} as a set with $|X| = |X^{-1}|$ and $X \cap X^{-1} = \emptyset$. We can create a bijection between the elements fo X and X^{-1}, and then for each $a \in X$ there is an $a^{-1} \in X^{-1}$, for each $a \in X^{-1}$ there is an $a^{-1} \in X$ and for each $a \in X \cup X^{-1}$ we have $a = (a^{-1})^{-1}$. We denote the set $X \cup X^{-1}$ as $X^{\pm}$.

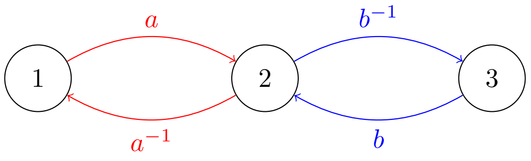
Definition 3.3 (Labeling)**.** *Given a graph Γ and a finite set $X^{\pm}$, a labeling is a function*

$$lab : E(\Gamma) \to X^{\pm}$$

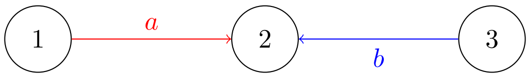
such that for each $e \in E(\Gamma)$, $lab(\overline{e}) = lab(e)^{-1}$.

Definition 3.4 (Labeled graph)**.** *Given a graph Γ and a labeling lab of this graph, we call the pair (Γ, lab) a labeled graph.*

The last equality of the labeling definition allows us to represent the labeled graph Γ using only the X-labeled edges, because the X^{-1}-labeled edges can be deduced immediately from them.



(a) Diagram with all the edges

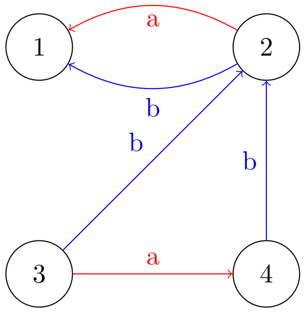


(b) Diagram with only the X-labeled edges

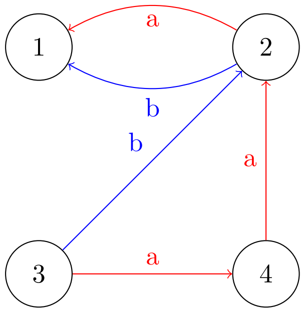
Figure 2: Both directed diagrams represent the same labeled graph

Definition 3.5 (Well-labeled graph)**.** *A labeled graph Γ is well-labeled if*

$$\iota(e_1) = \iota(e_2), lab(e_1) = lab(e_2) \Rightarrow e_1 = e_2$$



(a) Not well-labeled graph



(b) Well-labeled graph

Figure 3: The first graph is not well-labeled because there are two edges with $lab(e) = b^{-1}$ and $\iota(e) = 2$

Definition 3.6 (Path)**.** *A path p in a graph Γ is a sequence*

$$\{v_0, e_1, v_1, e_2, v_2, ..., e_{n-1}, v_{n-1}, e_n, v_n\}$$

such that $v_i \in V(\Gamma)$, $e_i \in E(\Gamma)$ and also $\iota(e_i) = v_{i-1}$ and $\tau(e_i) = v_i$. We call n the length of the path. We denote $\iota(p) = v_0$ and $\tau(p) = v_n$.

Definition 3.7 (Distance)**.** *Given two vertices of the graph Γ, $u, v \in V(\Gamma)$, we define the distance between u and v as the minimum length of a path p that has $\iota(p) = u$ and $\tau(p) = v$.*

Definition 3.8 (Path concatenation)**.** *If we have two paths p_1 and p_2 fo Γ such that $\tau(p_1) = \iota(p_2)$ we can concatenate both paths and obtain a new path*

$$p = p_1 p_2 = \{v_0^1, e_1^1, ..., e_{n_1}^1, v_{n_1}^1 = v_0^2, e_1^2, ..., v_{n_2}^2\}$$

Definition 3.9 (Reverse (inverse) path)**.** *Given a path $p = \{v_0, e_1, ..., v_n\}$, we define the reverse (inverse) path in Γ as the sequence*

$$\overline{p} = \{v_n, \overline{e_n}, v_{n-1}, \overline{e_{n-1}}, ..., v_1, \overline{e_1}, v_0\}$$

that is obviously a path because for each $e \in E(\Gamma)$ we have $\iota(e) = \tau(\overline{e})$.

Definition 3.10 (Closed path)**.** *A path p is closed if $\iota(p) = \tau(p)$.*

Definition 3.11 (Reduced path)**.** *A reduced path is a path where there is no i such that $e_i = \overline{e_{i+1}}$.*

Definition 3.12 (Word)**.** *Given a finite set $X^{\pm}$, a word on $X^{\pm}$ is the concatenation of a nonnegative number of elements of $X^{\pm}$.*

Definition 3.13 (Word length)**.** *If we have a word on $X^{\pm}$ $w = x_1x_2...x_n$, we say that w has length n, and we represent it by $|w| = n$.*

Definition 3.14 (Inverse word)**.** *If we have a word on $X^{\pm}$ $w = x_1x_2...x_n$, we call $w^{-1} = x_n^{-1}...x_2^{-1}x_1^{-1}$ the inverse word of w.*

Definition 3.15 (Freely reduced word)**.** *A word over a finite set $X^{\pm}$, $w = a_1a_2...a_n$, is called freely reduced if there is no i such that $a_i = a_{i+1}^{-1}$.*

Definition 3.16 (Path label)**.** *Given a path $p = \{v_0, e_1, ..., v_n\}$ on a labeled graph Γ, we define the label of p as the word:*

$$lab(p) = lab(e_1)lab(e_2)...lab(e_n) \in (X^{\pm})^*$$

Remark 3.17. *Given a path p on a labeled graph Γ, we have that $lab(p) = lab(\overline{p})^{-1}$.*

Definition 3.18 (Freely reduced path)**.** *Given a path p on a labeled graph Γ, we call p freely reduced if $lab(p)$ is a freely reduced word.*

Remark 3.19. *Let p be a path on the labeled graph Γ. If p is freely reduced, p is reduced. If Γ is well-labeled, p is freely reduced if and only if p is reduced.*

Definition 3.20 (Connected graph)**.** *A graph Γ is connected if for every pair of vertices $v, u \in V(\Gamma)$ there is at least one path p with $\iota(p) = v$ and $\tau(p) = u$.*

Definition 3.21 (Connected component)**.** *Given a graph Γ, we call a maximal connected subgraph a connected component of Γ.*

Definition 3.22 (Tree)**.** *We call a graph Γ a tree if for every pair of vertices $v, u \in V(\Gamma)$ there is exactly one reduced path p with $\iota(p) = v$ and $\tau(p) = u$.*

Definition 3.23 (Spanning tree)**.** *Let Γ be a connected graph, we call a spanning tree of Γ a subgraph $T \subseteq \Gamma$ that $V(T) = V(\Gamma)$ and T is a tree.*

Definition 3.24 (Pointed graph)**.** *A pointed graph is a graph Γ with a distinguished vertex $v_0 \in V(\Gamma)$ called the basepoint. We represent that as (Γ, v_0).*

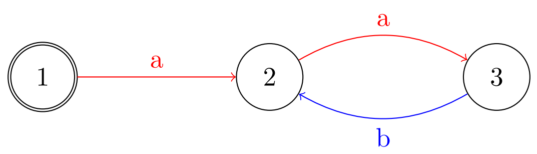


Figure 4: A pointed labeled graph with 1 as basepoint

We will represent the pointed graphs as normal graphs but with a double circle at the basepoint, as in Figure 4.
Now, given a group $G = \langle X \mid \mathcal{R} \rangle$, we know that the elements of G are equivalence classes of words over $X^{\pm}$. We will use the notation "$\equiv$" to say that two words are the same, and "$=_G$" to say that two words represent the same element of G.
Given a pointed labeled graph (Γ, v_0), we define $Loop(\Gamma, v_0)$ and $Lab(\Gamma, v_0)$ as follows:

$$Loop(\Gamma, v_0) = \{p \mid p \text{ is a path in } \Gamma \text{ with } \iota(p) = \tau(p) = v_0\}$$

$$Lab(\Gamma, v_0) = \{g \mid g \in G \text{ and there is } p \in Loop(\Gamma, v_0) \text{ with } lab(p) =_G g\}$$

Lemma 3.25. *$Lab(\Gamma, v_0) = H$, where H is a subgroup of G.*

Proof. It is obvious that $Lab(\Gamma, v_0) \subseteq G$, so we have to check that it is also a group. For each $p \in Loop(\Gamma, v_0)$ we have $\overline{p} \in Loop(\Gamma, v_0)$. And from the Definitions 3.9 and 3.4 we have that if $lab(p) =_G g$ then $\overline{p} =_G g^{-1}$, so $g \in H \Rightarrow g^{-1} \in H$. Obviously the path $\{v_0\} \in Loop(\Gamma, v_0)$, so $1 \in H$. For each pair $p_1, p_2 \in Loop(\Gamma, v_0)$ we have $\tau(p_1) = \iota(p_2) = v_0$ so we can concatenate them and obtain $p = p_1p_2$, with $lab(p) = lab(p_1)lab(p_2)$. If $lab(p_i) =_G g_i$, $lab(p) =_G g_1g_2$, so $g_1, g_2 \in H \Rightarrow g_1g_2 \in H$. □

Definition 3.26 (Core graph)**.** *Let Γ be a graph, the core of Γ at $v \in V(\Gamma)$ is the subgraph $Core(\Gamma, v)$ induced by the union of the reduced closed paths in Γ starting at v.*

It is easy to see that $Core(\Gamma, v)$ is a connected subgraph of Γ containing v. If $Core(\Gamma, v) = \Gamma$ we say that Γ is a core graph with respect to v.
The following lemma lists some properties of core graphs.

Lemma 3.27. *Let $\Gamma' = Core(\Gamma, v)$. Then:*

1. *Γ' is connected and contains the vertex v.*
2. *Γ' has no degree-one vertices, except possibly for the vertex v.*
3. *If Γ is a labeled graph then $Lab(\Gamma', v) = Lab(\Gamma, v)$.*

Proof. The path $p = \{v\}$ is obviously reduced and closed at v, so $v \in V(\Gamma')$. As each vertex $u \in V(\Gamma')$ is part of a closed path starting at v, there is one path from v to every other vertex of Γ', so it is connected.

Each vertex $u \in V(\Gamma')$ with $u \neq v$ is part of a reduced path p closed at v, so there are two edges $e_1, e_2 \in E(\Gamma')$ with $\iota(e_i) = u$ and $e_1 \neq e_2$.

As $\Gamma' \subseteq \Gamma$ we have easily that $Lab(\Gamma', v) \subseteq Lab(\Gamma, v)$. Let $p \in Loop(\Gamma, v)$, we call $p = p_1$. Now we build the sequence $p_1, p_2, ..., p_n$ where p_i is obtained from p_{i-1} after removing a continuous subsequence of the form $\{v_1, e, v_2, \overline{e}\}$. As p is finite, this sequence is also finite and p_n is obviously reduced and so $p_n \in Loop(\Gamma')$. Now, as $lab(e) = lab(\overline{e})^{-1}$ we have that, if $g_i =_G lab(p_i)$, $g_i = g_{i+1}$, and so $Lab(\Gamma, v) \subseteq Lab(\Gamma', v)$. □

Lemma 3.28. *Let $\Gamma' = Core(\Gamma, v)$, where (Γ, v) is a pointed graph. Then every reduced path p in Γ' such that $\iota(p) = v$ can be extended with another path p' such that $\tau(p') = v$ and pp' is reduced.*

Proof. If $\tau(p) = v$ we are done. Let $\tau(p) = u$, from the definition of Core graph we have that there is a reduced closed path at v that goes through u, $t = t_1t_2$, where $\tau(t_1) = u$. As t is reduced, the first edge of $\overline{t_1}$ and the first of t_2 cannot be the same, so at least one of them is not equal to the first edge of $\overline{p}$, let us call this path, $\overline{t_1}$ or t_2, t', then pt' is a reduced closed path at v. □

Definition 3.29 (Morphism of labeled graphs)**.** *Let Γ and Δ be two $X^{\pm}$ labeled graphs. We call the map $\pi : \Gamma \to \Delta$ a morphism of labeled graphs if π sends vertices to vertices, edges to edges, preserves labels for directed edges and the following holds*

$$\iota(\pi(e)) = \pi(\iota(e)) \text{ and } \pi(\overline{e}) = \overline{\pi(e)}, \forall e \in E(\Gamma)$$

Definition 3.30 (Embedding)**.** *We call a morphism of labeled graphs an embedding if it is injective. If there is an embedding from Γ to Δ we say that Γ embeds Δ.*

Definition 3.31 (Morphism of pointed labeled graphs)**.** *Let (Γ, v_0) and (Δ, u_0) be two pointed $X^{\pm}$ labeled graphs. We call the map $\pi : \Gamma \to \Delta$ a morphsim of pointed labeled graphs if π is a morphism of labeled graphs and sends v_0 to u_0.*

Lemma 3.32. *Let (Γ, v_0) and (Δ, u_0) be two $X^{\pm}$ pointed labeled graphs and $H = Lab(\Gamma, v_0)$ and $K = Lab(\Delta, u_0)$ two subgroups of the group $G\langle X \mid R\rangle$. If there exists a morphism of pointed labeled graphs $\pi : \Gamma \to \Delta$, then $H \leq K$.*

Proof. Let p be a path in Γ, we have that $lab(p) \equiv lab(\pi(p))$ and if p is closed and $\iota(p) = v_0$ then $\pi(p)$ is closed and $\iota(\pi(p)) = u_0$. With this we have $Loop(\Gamma, v_0) \subseteq Loop(\Delta, u_0)$ so $H \leq K$. □

Lemma 3.33. *Let (Γ, v_0) and (Δ, u_0) be two $X^{\pm}$ connected pointed labeled graphs, if Δ is well-labeled there is at most one morphism of pointed labeled graphs $\pi : \Gamma \to \Delta$.*

Proof. We will use induction to prove that there is at most one way to map the vertices of Γ to the ones on Δ. We will do induction over the distance n from $v \in V(\Gamma)$ to v_0. When $n = 0$ we have $v = v_0$ so $\pi(v_0) = u_0$ always. Now, suppose that there is only one way to map the vertices with distance less than $n > 0$, if $v \in V(\Gamma)$ has distance to v_0 equal to n there is a path p of length n with $\iota(p) = v_0$ and $\tau(p) = v$. Let v_{n-1} be the vertex just before v in this path, this vertex has distance to v_0 less than n, so there is only one way to map it. If e_n is the last edge of the path p, we have that there is only one edge $e \in E(\Delta)$ with $\iota(e) = \pi(v_{n-1})$ and $lab(e) = lab(e_n)$, so v con only be mapped to $\pi(v) = \tau(e)$. □

Remark 3.34. *Let (Γ, v_0) and (Δ, u_0) be two $X^{\pm}$ pointed well-labeled graphs, if they are isomorphic there is only one isomorphism $\pi : \Gamma \to \Delta$. We denote it $(\Gamma, v_0) \cong (\Delta, u_0)$.*

Definition 3.35 (Labeled product graph)**.** *Given two labeled graphs Γ and Δ, we define the labeled product graph as the graph $\Gamma \times \Delta$, with $V(\Gamma \times \Delta) = V(\Gamma) \times V(\Delta)$. We define $E(\Gamma \times \Delta)$ as*

$$\{e_1 \times e_2 \mid e_1 \in E(\Gamma), e_2 \in E(\Delta), lab(e_1) = lab(e_2)\}$$

with the following definitions: $\iota(e_1 \times e_2) = \iota(e_1) \times \iota(e_2)$, $\overline{e_1 \times e_2} = \overline{e_1} \times \overline{e_2}$ and $lab(e_1 \times e_2) = lab(e_1) = lab(e_2)$.

Definition 3.36 (Pointed labeled product graph)**.** *Given two pointed labeled graphs (Γ, v_0) and (Δ, u_0), we define the pointed labeled product graphs as $(\Gamma \times \Delta, v_0 \times u_0)$, were $\Gamma \times \Delta$ is the labeled product graph of Γ and Δ.*

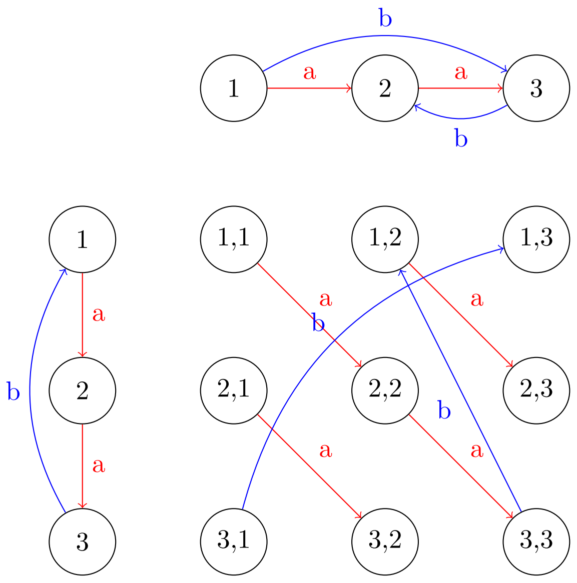


Figure 5: A labeled product graph

Lemma 3.37. *Given two well-labeled graphs Γ and Δ, then $\Gamma \times \Delta$ is also well-labeled.*

Proof. Suppose that $\Gamma \times \Delta$ is not well-labeled, there exists one vertex $u \times v$ with two edges $e^1 = e_1^1 \times e_2^1$ and $e^2 = e_1^2 \times e_2^2$ with $\iota(e^1) = \iota(e^2) = u \times v$, $lab(e^1) = lab(e^2)$ and $e^1 \neq e^2$. This implies that at least one of these holds: $e_1^1 \neq e_1^2$ or $e_2^1 \neq e_2^2$, from this we get that at least one of Γ or Δ is not well-labeled. □

Proposition 3.38. *Given two connected graphs* Γ *and* Δ*, With* $E(\Gamma) = n$ *and* $E(\Delta) = m$*, we can compute* $\Gamma \times \Delta$ *in* $O(nm)$*.*

Proof. As Γ and Δ are connected $V(\Gamma)$ is $O(n)$ and $V(\Delta)$ is $O(m)$. Then $V(\Gamma \times \Delta)$ is $O(nm)$. For each pair of edges from Γ and Δ we have to check if they have the same label, and hence they correspond to an edge in $\Gamma \times \Delta$. There are exactly $n \cdot m$ pairs so $E(\Gamma \times \Delta)$ is $O(nm)$. □

4 Data structures and methods

In this section we will describe, explain and analyze some data structures and methods that will be useful in the future for an efficient implementation of the algorithms. All the algorithms and methods we will be describing in this section and further information about them can be found in [3].

4.1 Linked list

The first and most basic data structure we will be using is the linked list. This data structure allows us to store elements with insertion and deletion operations that run in $O(1)$.
In a linked list we will have two data types, the list base and the list elements. The base element has a pointer `next` pointing to the first element of the list, if the list is empty the pointer is empty to. Each list element has two pointers, `next` and `prev`. As the names suggest, the `next` pointer points to the next element on the list, or it is empty if the element is the last one, and the `prev` pointer points to the previous element of the list or to the base list if it is the first element.
In the example of the Figure 6 we can see that `a.prev` is the list base because `a` is the first element and `e.next` is empty because `e` is the last element.

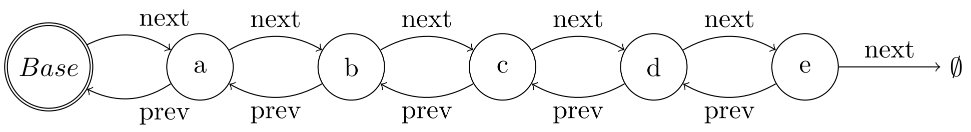


Figure 6: A linked list with five elements

Every time that we will work with lists or use lists in a method or algorithm, we will be working with the list base. So when we say that we pass a list as argument in some method or that some element is pointing to a list, we are talking about the list base.

Now we can easily describe insertion and deletion methods, note that as each element has information from the previous and next element we can delete it preserving the relative order of the other elements. Also, as we alway can work with the list base, we can always store the new elements at the beginning of the list an preserve the order of the other elements.

Method 1: Insert new element to a list

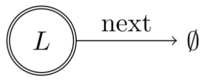
```
Input  : L – linked list (list base)
         x – list element we want to insert
1 Function INSERT(L,x) is
2     if $L.next \neq \emptyset$ then
3         x.next ← L.next
4         L.next.prev ← x
5     end
6     x.prev ← L
7     L.next ← x
8 end
```

The basic two methods for a linked list will be insert and delete, the first one will need the list and the new element we want to insert and the second one only the element we want to delete.

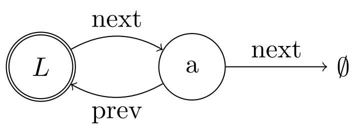
Method 2: Delete element from a list

```
Input  : x – list element we want to delete
Function DELETE(x) is
    if $x.next \neq \emptyset$ then
        x.next.prev ← x.prev
    end
    x.prev.next ← x.next
end
```

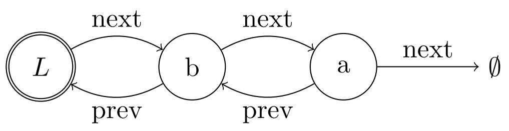
We can see easily that both methods run in $O(1)$ time. In Figure 7 we can see an example of how the list change when we insert or delete some elements.
As we will see in future sections, list elements or list bases can have more pointers or parameters, but the only ones that will interfere with the lists methods are `next` and `prev`, the last one only for list elements.



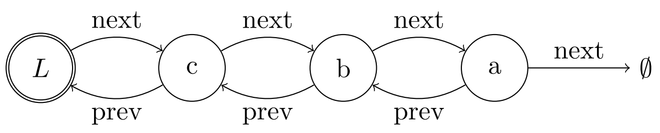
(a) Empty list



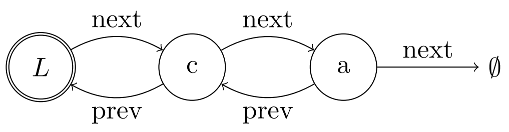
(b) List L after `INSERT`(L, a)



(c) List L after `INSERT`(L, b)



(d) List L after `INSERT`(L, c)



(e) List L after `DELETE`(b)

Figure 7: Example of how the linked list changes after the methods application

4.2 Queue

A queue is a data structure that works with the first in, first out concept. That means that we can do to operations, add one element to the queue or take one element from the queue. Every time we take one element we are taking the first element that has been inserted from the elements that are at this moment in the queue.
We will implement this data structure using a small modification of the linked list we described

before. The difference will be that the base element will also have a `prev` pointer pointing to the las element, an last elements `next` will point to the base point. Initially, when the queue is empty, the base element will be pointing to himself with `next` and `prev`. With this modification we have to change the insert method a bit, and the take method will delete and return the last element of the list.
For convention we will call the queue insert method `PUSH` and the take method `POP`, this will help to differentiate them from the list ones. We have that both methods have $O(1)$ time complexity.

Method 3: Insert new element to a queue

```
   Input  : Q – Queue (queue base)
            x – queue element we want to insert
1  Function PUSH(Q,x) is
2      x.next ← Q.next
3      Q.next.prev ← x
4      x.prev ← Q
5      Q.next ← x
6  end
```

Method 4: Take the last oldest from a queue

```
   Input  : Q – Nonempty queue
   Output: The oldest element in insertion order that is in the queue
1  Function POP(Q) is
2      x ← Q.prev
3      x.prev.next ← Q
4      Q.prev ← x.prev
5      x.next ← ∅
6      x.prev ← ∅
7      return x
8  end
```

4.3 Disjoint sets

Given a set of elements, this data structure allows us to maintain information about disjoint subsets of this elements and we can also merge two subsets in one in an efficient way. Each of the elements has a pointer `parent` that initially points to himself and and integer `rank` that is initially set to 0. This data structure has two basic methods, the first one is to find the representative of the element's subset. This representative is always an element of the subset and only changes when we merge two subsets, that one of the representatives become the representative of the new subset. This first method is called `FIND-SET`. The second method is `UNION`, and as the name says it unifies two subsets in one new subset. We just need one element from each subset, is not necessary that they are the representatives.

Method 5: Finds the representative of the subset of x

```
   Input  : x – set element
   Output: The representative of the subset of x
1  Function FIND-SET(x) is
2      if x.parent = x then
3          return x
4      else
5          x.parent ← FIND-SET(x.parent)
6          return x.parent
7      end
8  end
```

Note that initally each element belongs to its own subset, so each element is the representative of its own subset.

```
Method 6: Merges the subsets of x and y
Input  : x – set element
         y – set element
1  Function UNION(x,y) is
2      x ← FIND-SET(x)
3      y ← FIND-SET(y)
4      if $x \neq y$ then
5          if $x.rank > y.rank$ then
6              y.parent ← x
7          else
8              if $x.rank = y.rank$ then
9                  y.rank ← y.rank + 1
10             end
11             x.$parent$ ← y
12         end
13     end
14 end
```

In [1] we can find a proof that this data structure works in $O(m\ log^{*}n)$ amortized time complexity, where n is the number of elements in the set and m the numbers of times we call the methods.

Definition 4.1. *The function* $log^{*} : \mathbb{N} \to \mathbb{N}$ *assigns to each natural namber* n *the least natural number* k *such that:*

$$\underbrace{log \circ log \circ ... \circ log}_{k\ times}(n) \leq 1$$

From the definition of log^{*} we can see that this function is growing really slowly. If we take the base-2 logarithm we have that $log^{*}(2^{65536}) = 5$, where 2^{65536} is an integer much more larger than the bits of storage space of a normal computer. So, in practical sense we can consider this data structure as amortized linear in the number of times we use the methods.

4.4 Vertices and edges

Most of the time, in our algorithms, we are going to be working with vertices and edges of labeled graphs. That is way we need an efficient way to represent this concepts.
As we noticed in the previous section, we only need the directed X-labeled edges to represent a $X^{\pm}$ labeled graph. That is way we are going to consider as the same element the X-labeled edge e and the corresponding X^{-1}-labeled edge $\overline{e}$ when we do operation such as modifying some edges endpoints, adding or deleting an edge. So when we modify in some way the edge e we will also modify the edge $\overline{e}$.
To do that, each vertex v is going to have a linked list called `edges` with all the edges e such that $\iota(e) = v$. It is easy to see that each edge it will be at exactly one linked list. We will also assume that we have a pointer from e to $\overline{e}$, so that when we modify one of the edges the other one gets modified too.

5 Free group

We will first present the free group and some properties, all we are going to present about the free group is based on [5].
Informally, a group is free on a set of generators if no relation holds among these generators except the trivial relations that hold among any set of elements in any group.

Definition 5.1 (Free group)**.** *Let X be a subset of a group F. Then F is a free group with basis X if the following holds: if ϕ is any function from the set X into a group H, then there exists a unique extension of ϕ to a homomorphism ϕ^* from F to H.*

Remark 5.2. *The requirement that the extension be unique is equivalent to requiring that X generates F.*

We can now prove one of the basic properties of free groups.

Proposition 5.3. *Let F_1 and F_2 be free groups with bases X_1 and X_2. Then F_1 is isomorphic to F_2 if and only if X_1 and X_2 have the same cardinal.*

Proof. Suppose that f_1 is a one-to-one correspondence mapping X_1 onto X_2, and let $f_2 = f_1^{-1}$. The maps f_1 and f_2 determine maps $\phi_1 : X_1 \to F_2$ and $\phi_2 : X_2 \to F_1$. These have extensions to homomorphisms $\phi_1^* : F_1 \to F_2$ and $\phi_2^* : F_2 \to F_1$. Now $\phi_2^*\phi_1^* : F_1 \to F_1$ acts as the identity $f_2 f_1 = i_{X_1}$ on X_1, and hence is an extension of the inclusion map $X_1 \to F_1$. Since the identity $i_{F_1} : F_1 \to F_1$ also extends this inclusion map, by uniqueness we have $\phi_2^*\phi_1^* = i_{F_1}$. Similarly $\phi_1^*\phi_2^* = i_{F_2}$. It follows then that ϕ_1^* is an isomorphism from F_1 onto F_2.
It remains to show that F determines $|X|$. The subgroup N of F generated by all squares of elements in F is normal. This is because if w^2 is an element of the generating set for N, and $x \in F$, then $xw^2x^{-1} = (xwx^{-1})^2 \in N$, so for every $x \in F$ $xNx^{-1} \subseteq N$, hence N is normal. Also F/N is an elementary abelian 2-group. This is because every nontrivial element fo F/N has order 2. We have then that the elements of X form a basis for the vector space F/N over $\mathbb{F}_2$. So if X is finite $|F/N| = 2^{|X|}$ and if it is infinite $|F/N| = |X|$. □

Corollary 5.4. *All bases of a given free group F have the same cardinal, the rank of F.*

Now we will prove the existence of free groups in a constructive way. We will be using the set of words $(X^{\pm})^*$ together with a relation $\sim$. We first need to define an operation to get one word from another.

Definition 5.5 (Elementary transformation)**.** *Given a word on $X^{\pm}$ $w = x_1x_2...x_n$, we call an elementary transformation removing or adding a factor xx^{-1} with $x \in X^{\pm}$ to some position of w.*

With this definition we can now define the relation $\sim$ as: given two words $w_1, w_2 \in X^{\pm}$, we have $w_1 \sim w_2$ if an only if we can get from w_1 to w_2 using elementary transformations. Note that this is an equivalence relation and also holds that if $u_1 \sim u_2$ and $v_1 \sim v_2$ then $u_1v_1 \sim u_2v_2$, and also if $u_1 \sim u_2$ we have $u_1^{-1} \sim u_2^{-1}$.
Given a word $w \in (X^{\pm})^*$, we will denote the equivalence class of w in $(X^{\pm})^*/\sim$ by $[w]$. Now we will prove that we can use the freely reduced words to represent the elements of $(X^{\pm})^*/\sim$.

Proposition 5.6. *Given a finite set $X^{\pm}$, in each equivalence class of $(X^{\pm})^*/\sim$ there is exactly one freely reduced word.*

Proof. Given a word w, if we iteratively apply the removing elementary transformation we will get to one freely reduced word, so at $[x]$ there is at least one freely reduced word.
Now suppose that we have two freely reduced words $w \neq w'$ with $[w] = [w']$. That means that there is a sequence of words $\{w_1, w_2, ..., w_n\}$ with $w_1 = w$ ans $w_n = w'$ such that we can get from w_i to w_{i+1} with an elementary transformation. Let this sequence be the one with minimum $N = \sum_{i=1}^{n} |w_i|$, we obviously have that $|w_i| \neq |w_{i+1}|$ and that $n \geq 3$ because we cannot go from w to w' using only one elementary transformation. It must be true that $|w_1| < |w_2|$ and $|w_{n-1}| > |w_n|$, so there is one word w_i with $|w_i| > |w_{i-1}|, |w_{i+1}|$. This means that we are getting from w_i to w_{i+1} and w_{i-1} using a removing elementary transformation. If it is the same transformation it means that $w_{i+1} = w_{i-1}$ which contradicts the minimality of N. If they intersect, we have that w_i contains the subsequence $xx^{-1}x$ with $x \in X^{\pm}$ and that in both transformation the result of this subsequence is x, so again $w_{i+1} = wi - 1$. The last case is when they don't intersect, that means we remove a subsequence aa^{-1} to get from w_i to w_{i-1} and a subsequence bb^{-1} to get from w_i to w_{i+1}. We can replace w_i for w_i' consisting of w_i after removing both aa^{-1} and bb^{-1}, the sequence of words is still valid but with $N' = N - 4$ contradicting the minimality of N again. □

Given a word $w \in (X^{\pm})^*$, we will denote the only freely reduced word of $[w]$ by $\overline{w}$.

Proposition 5.7. *$F = (X^{\pm})^*/\sim$ is a free group with basis the set $[X]$ of equivalence classes of elements from X, and $|[X]| = |X|$. We will denote this group by $F(X)$.*

Proof. Let H be any group, and let ϕ map the set $[X]$ of equivalence classes $[x]$ of elements $x \in X$ into H. To show that $|[X]| = |X|$, we observe that if $x_1, x_2 \in X$ and $x_1 \neq x_2$, then $[x_1] \neq [x_2]$, since the two words are freely reduced. Then ϕ determines a map $\phi_1 : X \to H$ with $\phi([x]) = \phi_1(x)$. We can define an extension ϕ_1^* fo ϕ_1, from $(X^{\pm})^*$ into H as $\phi_1^*(w) = \phi_1^*(x_1^{e_1} x_2^{e_2} ... x_n^{e_n}) = \phi_1(x_1)^{e_1} \phi_1(x_2)^{e_2} ... \phi_1(x_n)^{e_n}$, with $x_i \in X$ and $e_i = \pm 1$. If w_1 and w_2 are equivalent, then $\phi_1^*(w_1) = \phi_1^*(w_2)$, whence ϕ_1^* maps equivalent words onto the same element of H. That induces a map $\phi^* : F \to H$ which is clearly a homomorphism and an extension of ϕ. As $[X]$ generates F, ϕ^* is unique. □

Corollary 5.8. *If X is any set, there exists a free group F with X as basis.*

Proposition 5.9. *If a group is generated by a set of n of its elements (n finite or infinite), then it is a quotient group of a free group of rank n.*

Proof. Let G be a group generated by $S \subseteq G$, with $|S| = n$. Let ϕ be a one-to-one map from a set X onto S. Let F be a free group with X as basis, then $\phi : X \to G$ extends to a homomorphism $\phi^* : F \to G$. Since the image S of X generates G, ϕ^* maps F onto G. □

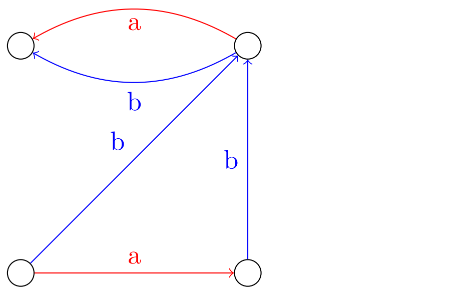
5.1 Stallings graphs

Here we are going to present the Stallings graphs, a geometric approach to study subgroups of free groups. This idea was first presented by Stallings in [9] and latter revised in a more combinatorial way by Kapovich and Myasnikov in [4].
The idea of Stallings graphs is to associate a unique pointed well-labeled core graph to each subgroup $H \leq_{fg} F(X)$, we will call it $\Gamma(H)$, that will allow us to solve problems like the word problem efficiently. In this section we will explain an algorithm to calculate efficiently $\Gamma(H)$ from a finite set of generators $H = \langle h_1, h_2, ..., h_m \rangle$ in time $O(n^2)$, where $n = \sum_{i=1}^{m} |h_i|$ and $|h|$ is the length of the freely reduced word on $X^{\pm}$ that represents h. There is a faster algorithm [10] that runs in $O(n \, log^* n)$, but the use of this algorithm instead of the other one won't change the final complexity of the malnormal closure calculation algorithm.

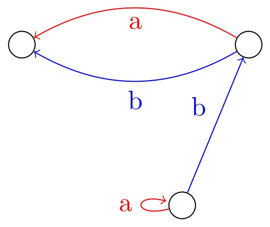
Definition 5.10 (Folding)**.** *Let Γ be a labeled graph but not well-labeled, then there is one vertex $v \in V(\Gamma)$ and two different edges $e_1, e_2 \in E(\Gamma)$ such that $\iota(e_i) = v$ and $lab(e_1) = lab(e_2)$. We call a folding of edges e_1 and e_2 the operation that gets Γ, e_1, e_2 and gives us the labeled graph Δ, where Δ is defined as follows:*

$$V(\Delta) = \{V(\Gamma) \setminus \{\tau(e_1), \tau(e_2)\}\} \cup \{u\}$$
$$E(\Delta) = \{f(e) \mid e \in E(\Gamma) \setminus \{e_1, e_2, \overline{e_1}, \overline{e_2}\}\} \cup \{e', \overline{e'}\}$$

Where f is a function $f : E(\Gamma) \to E(\Delta)$ such that $lab(f(e)) = lab(e)$, if $\iota(e) \in \{\tau(e_1), \tau(e_2)\}$ then $\iota(f(e)) = u$, else $\iota(e) = \iota(f(e))$. It holds that $\overline{f(e)} = f(\overline{e})$, so we define $\tau(f(e)) = \iota(\overline{f(e)})$. And we define e' and his inverse $\overline{e'}$ as $\iota(e') = v$, $lab(e') = lab(e_i)$ and $\iota(\overline{e'}) = u$.



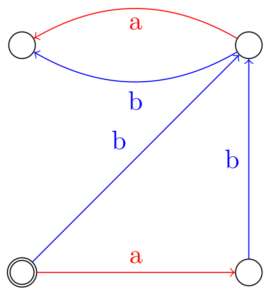
(a) Not well-labeled graph before Stallings folding



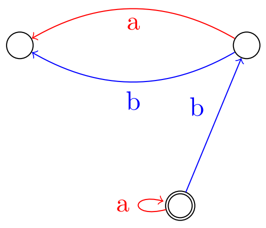
(b) Well-labeled graph after Stallings folding

Figure 8: We go from the first graph to the second doing a Stallings folding

We can see that if we have a labeled graph Γ and we fold edges $e_1, e_2 \in E(\Gamma)$, we get a graph Δ with $|E(\Delta)| + 1 = |E(\Gamma)|$ and $|V(\Gamma)| - 1 \leq |V(\Delta)| \leq |V(\Gamma)|$, where $|V(\Delta)| = |V(\Gamma)|$ only happens if $\tau(e_1) = \tau(e_2)$.
One thing to notice is that if we are doing a folding in a pointed labeled graph (Γ, v_0) and we get (Δ, u_0), then if $v_0 \in \{\tau(e_1), \tau(e_2)\}$ then $u_0 = u$, else $u_0 = v_0$.



(a) Not well-labeled pointed graph before Stallings folding



(b) Well-labeled pointed graph after Stallings folding

Figure 9: We go from the first graph to the second doing a Stallings folding, we can see how we change the basepoint

Proposition 5.11. *Let (Γ, v_0) be a pointed labeled graph. If we fold the edges $e_1, e_2 \in E(\Gamma)$ and we get the pointed labeled graph (Δ, u_0), then $Lab(\Gamma, v_0) = Lab(\Delta, u_0)$.*

Proof. There is an obvious morphism $f : (\Gamma, v_0) \to (\Delta, u_0)$, that for each vertex $v \in E(\Gamma) \setminus \{\tau(e_1), \tau(e_2)\}$ is the identity and $f(\tau(e_i)) = u$, this follows the definition of morphisms of pointed labeled graphs and, from Lemma 3.32, $Lab(\Gamma, v_0) \leq Lab(\Delta, u_0)$.
Let $p \in Loop(\Delta, u_0)$, Let us split p into the maximum number of paths $p = p_1 p_2 ... p_n$ such that if $i > 1$ then $\iota(p_i) = u$ and if $i < n$ then $\tau(p_i) = u$. As n is maximum, $u \notin p_i \setminus \{\iota(p_i), \tau(p_i)\}$ for all $0 \leq i \leq n$. So, for each i, there is a path p'_i from Γ such that $f(p'_i) = p_i$ and if $i > 1$ then $\iota(p'_i) \in \{\tau(e_1), \tau(e_2)\}$ and if $i < n$ then $\tau(p'_i) \in \{\tau(e_1), \tau(e_2)\}$. Now we can create a path $p' = p'_1 t_1 p'_2 t_2 ... t_{n-1} p'_n$ where $t_i = \{\tau(p'_i), e'_1, v, e'_2, \iota(p'_{i+1})\}$, having $v = \iota(e_i)$, $e'_1 \in \{\overline{e_1}, \overline{e_2}\}$ and $e'_2 \in \{e_1, e_2\}$ depending on the values of $\tau(p'_i)$ and $\iota(p'_{i+1})$. We have that p' is a path in Γ. It could happen that $p_1 = p_n = \{u\}$ if $v_0 \in \{\tau(e_1), \tau(e_2)\}$, then we choose $p'_1 = p'_n = \{v_0\}$. So now we have that $p' \in Loop(\Gamma, v_0)$ and $lab(t_i) =_G 1$ so $lab(p') =_G lab(p)$ which means $Lab(\Gamma, v_0) \geq Lab(\Delta, u_0)$. □

Proposition 5.12. *Let (Γ, v_0) be a pointed well-labeled graph. Then for each word $w \in lab(Loop(\Gamma, v_0))$ we have $\overline{w} \in lab(Loop(\Gamma, v_0))$.*

Proof. Let $p \in Loop(\Gamma, v_0)$, then as Γ is well-labeled we have that $lab(p)$ is freely reduced if and only if p is reduced from Remark 3.19. So, if we have the sequence $p_1, p_2, ..., p_n$ where $p_1 = p$, p_n is reduced, and we get p_i from p_{i-1} after removing two consecutive edges such that $e_j = \overline{e_{j+1}}$, then it is obvious that $p_i \in Loop(\Gamma, v_0)$ and also that $[lab(p_i)] = [lab(p)]$. And as p_n is reduced $lab(p_n) = \overline{lab(p)}$. □

We now can present a method that given a labeled graph Γ it will check if it is well-labeled, and if it is not, it will return two edges to fold.
As we said on Section 4.4, we are going to modify the edges e and $\overline{e}$ together and each vertex will have a linked list (explained in Section 4.1) called edges to store the edges with $\iota(e) = v$. For this purpose we are going to have a disjoint set data structure (explained in Section 4.3) to store the equivalence classes of vertices.
We are going to use a memory storage that we will call `Label`, it will have an entry for each element of $X^{\pm}$ and the direct access to each position will have $O(1)$ time complexity.

Method 7: Checks if the labeled graph Γ is well-labeled

```
Input  : $\Gamma$ – labeled graph
Output: $e_1$– edge to fold
        $e_2$– edge to fold
Function IS-WELL-LABELED($\Gamma$) is
    foreach $v \in V(\Gamma)$ do
        if $v = FIND\text{-}SET(v)$ then
            foreach $x \in X^{\pm}$ do
                $Label[x] \leftarrow \emptyset$
            end
            e ← v.edges.next
            while $e \neq \emptyset$ do
                if $Label[lab(e)] \neq \emptyset$ then
                    Result: $(Label[lab(e)], e)$
                end
                $Label[lab(e)] \leftarrow e$
                e ← e.next
            end
        end
    end
    Result: $\emptyset$
end
```

Now we need a method to merge the two edges found by the previous method. As before, we are representing the vertices with a disjoint set data structure to know which ones we still have. That means that every time we do a folding we are going to merge vertices $\tau(e_1)$ and $\tau(e_2)$ into the new vertex u. So we won't have to change the ι and τ values of any edge and we will only have to remove one of e_1 and e_2.

Now we have to prove that there exists a core pointed well-labeled graph for each subgroup $H \leq_{fg} F(X)$, also that this graph is unique and find an effective algorithm to calculate it.

Lemma 5.13. *Let (Γ, v) be a core pointed well-labeled graph that when related to $F(X)$ we have $Lab(\Gamma, v) = H$. For every prefix w of a reduced word representing an element $h \in H$ there is a unique path p in Γ with $\iota(p) = v$, and $lab(p) = w$.*

Proof. The uniqueness comes from the fact that the graph is well-labeled. And the existence is obvious if we take p to be the initial part of length $|w|$ of the closed path at v with label $\overline{h}$, whose existence follows from Proposition 5.12. □

Proposition 5.14. *Let $F(X)$ be the free group with finite basis X and let $K \leq H \leq F(X)$ be two subgroups of $F(X)$. Suppose (Γ_1, v_1) and (Γ_2, v_2) are core pointed well-labeled graphs such that $Lab(\Gamma_1, v_1) = K$ and $Lab(\Gamma_2, v_2) = H$.*
Then there exists a unique morphism of pointed labeled graphs $\phi : \Gamma_1 \to \Gamma_2$.

Proof. The uniqueness follows from Lemma 3.33. We have to prove that such π exists.
As Γ_1 is a core graph, every vertex in $V(\Gamma_1)$ is part of a reduced path. Let w be the prefix of a reduced word representing an element $h \in K$. From Lemma 5.13 we have that there is a path p_1 in Γ_1 with $\iota(p_1) = v_1$ and $lab(p_1) = w$, and also a path p_2 in Γ_2 with $\iota(p_2) = v_2$ and $lab(p_2) = w$. We define $\phi(\tau(p_1)) = \tau(p_2)$, and the image of the edges as the only possible edge following the morphism conditions as graphs are well-labeled, we need to check that this is well defined.
For each vertex of Γ_1 we will consider w as the shortest word holding the conditions and we will be doing induction over the length of the word. The base case is the basepoint which is obviously true.
Suppose we take another prefix $w' \neq w$ of a reduced word representing an element of K such that the path p_1' in Γ_1 with $\iota(p_1') = v_1$ and $lab(p_1') = w'$ has $\tau(p_1') = \tau(p_1)$. We will see that the

equivalent path p'_2 in Γ_2 has $\tau(p'_2) = \tau(p_2)$. As p_1 and p'_1 are freely reduced paths, we have that $lab(p_1\overline{p'_1}) \neq_{F(X)} 1$. Now we have two options, if the last edges of p_1 and p'_1 are the same, let u be the last edge in p_1 and p'_1 before $\tau(p_1)$, then $\phi(\tau(p_1))$ is well defined if $\phi(u)$ is well defined, that holds for the induction assumption. So let us suppose that the last edge is not the same. Then $p_1\overline{p'_1}$ is a reduced path, hence is freely reduced. So $lab(p_1\overline{p'_1}) \in K$. From Proposition 5.12 we have that there is a closed at v_2 reduced path p in Γ_2 with $lab(p) = lab(p_1\overline{p'_1}) = lab(p_2\overline{p'_2})$, so we have $\tau(p_2) = \tau(p'_2)$. □

Method 8: Folds two edges on the pointed labeled graph (Γ, v_0)

```
Input : $\Gamma$ – labeled graph
        $v_0$– basepoint
        $e_1$– edge to fold
        $e_2$– edge to fold
1  Function FOLD($\Gamma$,$v_0$,$e_1$,$e_2$) is
2      $u_1 \leftarrow$ FIND-SET($\tau(e_1)$)
3      $u_2 \leftarrow$ FIND-SET($\tau(e_2)$)
4      UNION($u_1, u_2$)
5      u $\leftarrow$ FIND-SET($u_1$)
6      if $v_0 \in \{u_1, u_2\}$ then
7          $v_0 \leftarrow$ u
8      end
9      if $u = u_1$ then
10         foreach $e \in u_2.edges$ do
11             DELETE(e)
12             if $e \neq e_2$ then
13                 INSERT(u.edges, e)
14             end
15         end
16     else
17         foreach $e \in u_1.edges$ do
18             DELETE(e)
19             if $e \neq e_1$ then
20                 INSERT(u.edges, e)
21             end
22         end
23     end
24 end
```

Proposition 5.15. *Let $F(X)$ be a free group with finite basis X. Let $H \leq_{fg} F(X)$ be a subgroup of $F(X)$. Then there exists a core pointed well-labeled graph (Γ, v_0) such that $Lab(\Gamma, v_0) = H$.*

Proof. Let us consider the labeled graph Δ where we have a bijection f between right cosets of H in $F(X)$ and the elements of $V(\Delta)$. The edge e with $\iota(e) = v_1$, $\tau(e) = v_2$ and $lab(e) = x$ with $v_1, v_2 \in V(\Delta)$ and $x \in X^{\pm}$ is in $E(\Delta)$ if and only if $f^{-1}(v_1)x = f^{-1}(v_2)$, where $f^{-1}(v_i)$ is the right coset assigned to each vertex.
This graph is connected. Let us call $v_0 = f(H) \in V(\Delta)$, for each vertex $v_1 = f(Hh)$, with $h \in F(X)$, there is a reduced path p with $lab(p) = \overline{h}$, $\iota(p) = v_0$ and $\tau(p) = v_1$.
Δ is well-labeled. Suppose we have $e_1, e_2 \in E(\Delta)$ with $lab(e_1) = lab(e_2) = x$ and $\iota(e_1) = \iota(e_2) = v$, and $u_i = \tau(e_i)$. Then we have $f^{-1}(u_1) = f^{-1}(v)x = f^{-1}(u_2)$, so $u_1 = u_2$ and hence $e_1 = e_2$.
Now let us see that $Lab(\Delta, v_0) = H$, from the definition of Δ it is obvious that $Lab(\Delta, v_0) \leq H$ because every element $h \in F(X)$ such that $Hh = H$ must satisfy $h \in H$. Let $h \in H$ and p a reduced path such that $\iota(p) = v_0$ and $lab(p) = \overline{h}$, if $\tau(p) \neq v_0$ we have that $H = Hh = Hk$, where $k \in F(X) \setminus H$, which is impossible. So $Lab(\Delta, v_0) \supseteq H$.

If we define $(\Gamma, v_0) = Core(\Delta, v_0)$, from Lemma 3.27 we have that this graph holds all the conditions. $\square$

Proposition 5.16. *Let $F(X)$ be a free group with finite basis X and let $H \leq_{fg} F(X)$. Suppose (Γ_1, v_1) and (Γ_2, v_2) are connected core pointed well-labeled graphs and $Lab(\Gamma_1, v_1) = H = Lab(\Gamma_2, v_2)$. Then there exists a unique isomorphism of pointed labeled graphs $\pi : \Gamma_1 \to \Gamma_2$.*

Proof. The uniqueness of π follows from Lemma 3.33.
Since $H \leq H$, by Proposition 5.14 there is a morphism of pointed graphs $\pi : (\Gamma_1, v_1) \to (\Gamma_2, v_2)$. We claim that π is an isomorphism of pointed graphs. Proposition 5.14 implies there is a morphism of pointed graphs $\pi' : (\Gamma_2, v_2) \to (\Gamma_1, v_1)$. Therefore $\pi' \circ \pi : (\Gamma_1, v_1) \to (\Gamma_1, v_1)$ is a morphism, from Lemma 3.33 it is unique, namely the identity map on (Γ_1, v_1). We can use a symmetric argument to show that $\pi \circ \pi'$ is the identity map on (Γ_2, v_2). Therefore π is an isomorphism. $\square$

Definition 5.17. *Given a finite set X and a subgroup $H \leq_{fg} F(X)$, we call the unique pointed well-labeled core graph (Δ, v_0) such that $Lab(\Delta, v_0) = H$, the Stallings graph of the subgroup H, and we denote it by $\Gamma(H)$.*

Using the Stallings graph $\Gamma(H)$ we can find a basis of the subgroup H efficiently. For that we will use a spanning tree T of the graph $\Gamma(H)$. we will denote by $[v, u]_T$ the only reduced path from v to u in T.

Proposition 5.18. *Given a subgroup $H \leq_{fg} F(X)$, let $T \subseteq \Gamma(H)$ be a spanning tree. Let v_0 be the basepoint of the pointed graph $\Gamma(H)$. We define the sets:*

$$E_T^+ = \{e \mid e \in E(\Gamma(H)) \setminus E(T) \text{ and } lab(e) \in X\}$$

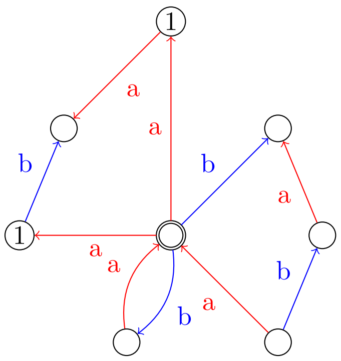
$$Y_T = \{p_e \in F(X) \mid lab([v_0, \iota(e)]_T e[\tau(e), v_0]_T) =_{F(X)} p_e \text{ with } e \in E_T^+\}$$

We have that $\langle Y_T \rangle = H$.

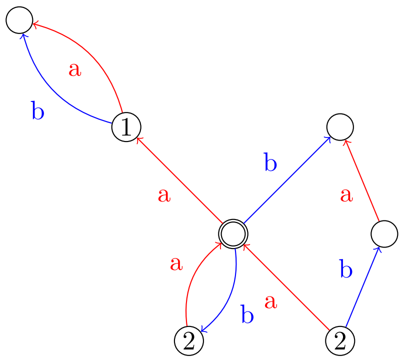
Proof. We can extend the definition of Y_T to $\overline{Y_T}$ using all the edges $e \in E(\Gamma(H))$ instead of only the ones if E_T^+. We can see that $Y_T \subseteq \overline{Y_T}$. It is easy to see that $p_{\overline{e}} = p_e^{-1}$, so we will only consider the elements $p_e \in \overline{Y_T}$ with $lab(e) \in X^+$.
It is obvious that $\langle Y_T \rangle \leq H$, so we will prove that every element h can be expressed as a product of elements from Y_T. Given an element $h \in H$ there is a word w in $(X^{\pm})^*$ such that $lab(w) =_{F(X)} h$ and w is freely reduced. From Proposition 5.12 we have that there is path p closed at v_0 with $w = lab(p)$. If $e_1, \ldots, e_n$ is the sequence of edges in the path p, we will define $u_i^{\delta_i}$ as $u_i = p_{e_i}$ and $\delta_i = 1$ if $lab(e_i) \in X^+$ or $u_i = p_{\overline{e_i}}$ and $\delta_i = -1$ in the other case. We have that $u_1^{\delta_1} u_2^{\delta_2} \ldots u_n^{\delta_n} = h$. All the $u_i \in \overline{Y_T}$, we can see that if $u_i \notin Y_T$ then $u_i = 1$, so h is a product of elements of Y_T. $\square$

Theorem 5.19. *Let $\{h_1, h_2, \ldots, h_m\}$ be a generator set for the subgroup $H \leq_{fg} F(X)$. There is an algorithm that builds $\Gamma(H)$ in $O(n^2)$, where $n = \sum_{i=1}^m |h_i|$.*

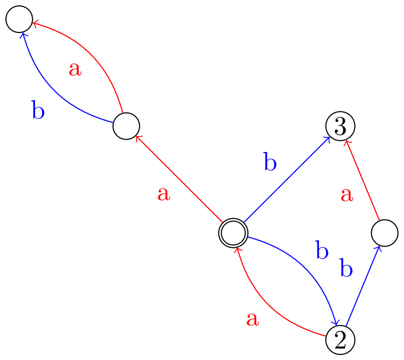
Proof. First we will create the called Flower graph $\mathcal{F}(H)$, it is a pointed graph with disjoint cycles that represent a closed path at v_0 with label h_i, one cycle for each. It is easy to see that $Lab(\mathcal{F}(H)) = H$. Now, we will keep applying Method 7 and Method 8 until we get a pointed graph that is well-labeled, we will call this graph $G(H)$. $G(H)$ is a pointed well-labeled graph such that $Lab(G(H)) = H$, by Proposition 5.11. If we iteratively remove the vertices with degree 1 we will end up with a core well-labeled pointed graph $G'(H)$ such that $Lab(G'(H)) = H$. By Proposition 5.16 we have that $G'(H) = \Gamma(H)$.
As Method 7 runs in $O(n)$ and Method 8 runs in $O(log^*(n))$ amortized time, and we at most use these methods $O(n)$ times, the total running time is $O(n^2)$. $\square$



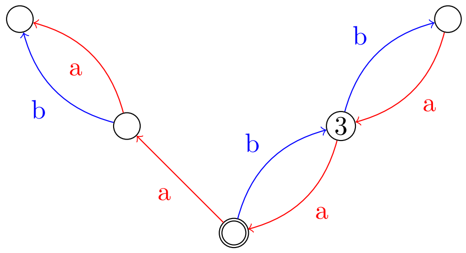
(a) The flower graph $\mathcal{F}(H)$ where $H = \langle aba^{-2}, ba^{-1}b^{-1}a, ba\rangle$



(b) The graph after folding the two edges with label a and terminal point at the vertices identified with 1



(c) The graph after folding the two edges with label a and initial point at the vertices identified with 2



(d) The graph after folding the two edges with label b and terminal point at the vertices identified with 2 and 3

Figure 10: The folding process from the flower graph to the Stallings graph

5.2 Malnormal closure algorithm

First we are going to see some relations between malnormality of a subgroup $H \leq_{fg} F(X)$ and some properties of $\Gamma(H)$. After that we will be able to present an algorithm that runs in $O(n^3)$ to get the malnormal closure. This algorithm was first presented in [8] by P. V. Silva and P. Weil.
The first proposition gives some conditions on $\Gamma(H) \times \Gamma(H)$ that are equivalent to H being a malnormal subgroup.
From now on, we will call the diagonal connected component of $\Gamma(H) \times \Gamma(H)$ the connected component that contains all the vertices (v, v) with $v \in V(\Gamma(H))$. It is easy to see that this subgraph is isomorphic to $\Gamma(H)$.

Proposition 5.20. *Let $H \leq_{fg} F(X)$, H is malnormal if and only if all the connected components of $\Gamma(H) \times \Gamma(H)$ except the diagonal are trees.*

Proof. Suppose that $\Gamma(H) \times \Gamma(H)$ has a connected component, which is not the diagonal, that has a reduced cycle. If (v_1, v_2) is a vertex of this component, $v_1 \neq v_2$ because it is not the diagonal component, we have a reduced nonempty path p closed at (v_1, v_2) with freely reduced label $w = lab(p)$. That means we can do the same path in $\Gamma(H)$ from v_1 to itself and the same for v_2. Let w_1 be the label of a reduced path from v_0 to v_1, and the same definition for w_2. Let g, g_1, g_2 represent the elements of H corresponding to w, w_1, w_2. We have $g \in g_1^{-1}Hg_1 \cap g_2^{-1}Hg_2$ which implies $g_2gg_2^{-1} \in H \cap g_2g_1^{-1}H(g_2g_1^{-1})^{-1}$, $1 \neq g_2gg_2^{-1}$ because w is a freely reduced nonempty word, and $g_2g_1^{-1} \notin H$ because $v_1 \neq v_2$, so H is not malnormal.
Now suppose H is not malnormal, then we have $g \notin H$, such that $H \cap gHg^{-1} \neq 1$. Let w be a reduced word representing g and $w = u\overline{v}$, where v is the longest prefix of $\overline{w}$ that we can get as a label of a path in $\Gamma(H)$ starting at v_0, ending at v_1. Then all the freely reduced labels of

closed paths at $\Gamma(gHg^{-1})$ are of the form $ut\overline{u}$, where t is a freely reduced nonempty word. We have $H \cap gHg^{-1} \neq 1$, so there is $h \neq 1$ with $u_H h u_H^{-1} \in H \cap gHg^{-1}$, where $u_H \in H$ is the element represented by the word u. If w_h is the freely reduced word representing h, we have that $uw_h\overline{u}$ is freely reduced, and there is a path p in $\Gamma(H)$ closed at v_0 such that $lab(p) = uw_h\overline{u}$. This means that there is a path p_u starting at v_0 in $\Gamma(H)$ that has $lab(p_u) = u$ and $\tau(p_u) = v_2$. If $v_1 = v_2$ we have that $g \in H$, so $v_1 \neq v_2$. It is easy to see that there are two closed reduced paths p_1, p_2 with $lab(p_i) = w_h$ and $\iota(p_i) = v_i$, so the connected component of $\Gamma(H) \times \Gamma(H)$ containing (v_1, v_2) has a reduced cycle and hence is not a tree. □

Corollary 5.21. *Let $H \leq_{fg} F(X)$, given $\Gamma(H)$ there is an algorithm that finds some $g \in F(X) \setminus H$ such that $gHg^{-1} \cap H \neq 1$ or determines that H is a malnormal subgroup of $F(X)$ in $O(n^2)$ time.*

Proof. As seen in Theorem 5.19 and Proposition 3.38 we can compute $\Gamma(H)$ and $\Gamma(H) \times \Gamma(H)$ in $O(n^2)$. We can check for cycles in $\Gamma(H) \times \Gamma(H)$ except the diagonal component in $O(n^2)$. If there is no cycle, H is malnormal. If there is one, we take a vertex (v_1, v_2) from the component that has the cycle. Let u and v be the words from the proof of Proposition 5.20, we can find them in $O(n^2)$ time and then the word $w = u\overline{v}$ represents an element $g \in F(X) \setminus H$ such that $gHg^{-1} \cap H \neq 1$. □

Lemma 5.22. *Let $H \leq G$. If $g \in G \setminus H$ satisfies $gHg^{-1} \cap H \neq 1$ then $g \in H_{mal}$.*

Proof. Suppose that $g \notin H_{mal}$, then $g \in G \setminus H_{mal}$. But $1 \neq gHg^{-1} \cap H \leq gH_{mal}g^{-1} \cap H_{mal} = 1$ leads to a contradiction, so $g \in H_{mal}$. □

Now we can prove the first proposition showing that the malnormal closure computation for finitely generated subgroups of free groups has a polynomial solution. This proposition was first proved by P. V. Silva and P. Weil in [8].

Proposition 5.23. *Let $\langle h_1, \ldots, h_m \rangle = H \leq_{fg} F(X)$, we can compute H_{mal} in $O(n^3)$. Where $n = \sum_{i=1}^{m} |h_i|$.*

Proof. We will prove that we can get a sequence $H = H_0 \leq H_1 \leq ... \leq H_k = H_{mal}$, with $k \leq n$ and that we can get $\Gamma(H_{i+1})$ from $\Gamma(H_i)$ in $O(n^2)$.
As we saw in Theorem 5.19 we can construct $\Gamma(H)$ in $O(n^2)$. Suppose now we have $\Gamma(H_i)$, as Corollary 5.21 says, we can check if H_i is malnormal or find an element g that contradicts the malnormal definition in $O({n_i}^2)$. If H_i is malnormal, we are done and $H_{mal} = H_i$. If not, as in the proof of Corollary 5.21, we can find $g =_{F(X)} u\overline{v}$ such that v and u can be read in $\Gamma(H_i)$ from v_0, ending at q_v and q_u with $q_v \neq q_u$ because $g \notin H_i$. Now, as we saw in Lemma 5.22, $\langle g, H_i \rangle \leq H_{mal}$, because $H_i \leq H_{mal}$. We define $H_{i+1} = \langle g, H_i \rangle$, $\Gamma(H_{i+1})$ can be computed from $\Gamma(H_i)$ by identifying vertices q_v and q_u and doing the necessary foldings, this process takes $O({n_i}^2)$ time.
We have seen that each step takes $O({n_i}^2)$, but also if n_i is the number of edges of $\Gamma(H_i)$, $n_{i+1} \leq n_i$, so each of the k steps runs in $O(n^2)$. But $\Gamma(H_{i+1})$ has at least one vertex less than $\Gamma(H_i)$, so $k \leq n$ as we wanted. □

5.2.1 Product Stallings foldings

In this section we are going to present the product Stallings foldings, these foldings will allow us to obtain $\Gamma(H') \times \Gamma(H')$ from $\Gamma(H) \times \Gamma(H)$ efficiently if $\Gamma(H')$ is obtained from $\Gamma(H)$ by a series of vertex identifications and the consequent Stallings foldings. We will first describe the product Stallings folding operation, then we will describe the product vertex identification and prove the correctness of both operations.
From now on, we will be working with well-labeled pointed graphs such as Γ with the property that $V(\Gamma) = Q \times Q$, where Q is a finite set. Given $p, q \in Q$, with $p \neq q$, we will define the Product Stallings folding of elements p and q as $\mathcal{PSF}(p, q)$.
This operation will give us a new graph Γ', with $V(\Gamma') = Q' \times Q'$, where $Q' = Q \setminus \{q\}$. This graph will be obtained from Γ after merging the vertices (q, t) to (p, t) and (t, q) to (t, p), with $t \in Q \setminus \{p, q\}$, and merging $(p, q), (q, p), (q, q)$ to (p, p). We can see that the first two types of merges are symmetric, and that the last one can be explained in three simple merges: (q, q) to (p, q), (q, p) to (p, p) and (p, q) to (p, p). We will use this symmetry to explain only the merges of

vertices (q,t) to (p,t), and the rest follows from this definition.
The intuition for the merging of (q,t) to (p,t) is that during all this section $\Gamma \subseteq \Delta \times \Delta$, where Δ is a labeled, not necessary well-labeled, pointed graph with $V(\Delta) = Q$. When we are performing $\mathcal{PSF}(p,q)$ we are getting $\Gamma' \subseteq \Delta' \times \Delta'$ where Δ' is obtained from Δ after performing a normal Stallings folding that ends up merging the vertices p and q. $E(\Gamma') \subseteq E(\Delta' \times \Delta')$ is such that Γ' is well-labeled.
As we said in previous sections, when we work with edges doing an operation such as deleting, crating or changing endpoints, we are in fact applying this operation to the edge e and to the corresponding edge $\overline{e}$.

Now we can explain how the merge of (q,t) to (p,t) will work. As they will end up being the same vertex, we have to move all the edges $e \in E(\Gamma)$ with $\iota(e) = (q,t)$ to a new edge e' with $\tau(e) = \tau(e')$ and $lab(e) = lab(e')$ and $\iota(e') = (p,t)$. In some cases it can happen that we already have an edge e_1 with $\iota(e_1) = (p,t)$ and $lab(e_1) = lab(e)$. In this case we can see that in the graph Δ, after identifying vertices q,p, let $\tau(e) = (k_1^1, k_2^1)$ and $\tau(e_1) = (k_1^2, k_2^2)$, if we apply the Stallings folding algorithm until Δ is well-labeled we will end up identifying k_i^1 with k_i^2. So in this case we store that we have to identify the pairs (k_i^1, k_i^2) and remove the edge e because we already have an edge that at the end of the algorithm will be equivalent.
We have a special case if we want to move an edge $e \in E(\Gamma)$ with $\iota(e) = (q,t) = \tau(e)$. In this case we have to check if (p,t) already has edges with $lab(e') = lab(e)$ or $lab(e') = lab(\overline{e})$. If none of these edges exits we move e from (q,t) to (p,t). If some or both of this edges exists, we remove e and we will store that we have to identify (p, k_1^i) and (t, k_2^i), where (k_1^i, k_2^i) where $i = 1$ is the end point of the edge $\iota(e') = (p,t)$, $lab(e') = lab(e)$, and $i = 2$ is the end point of the edge $\iota(e'') = (p,t)$, $lab(e'') = lab(\overline{e})$, if they exist.

As we did with the Stallings folding algorithm we are going to present the method to perform the Product Stallings folding. In Method 8 we had as input the pointed graph and the two edges we wanted to fold, in this case, as we already presented, we need the graph Γ and the two vertices $p, q \in V(\Delta)$ that we are identifying, we also will need the basepoint of Δ, v_0, to maintain it. As we did with the Stallings folding, we will use the data structure for disjoint sets, explained in Section 4.3, to keep the actual set of vertices of Δ. We will also have a queue Q where we will store the future pairs of vertices that we have to identify, queues are explained in Section 4.2. In addition we will have S, that is a list of the remaining vertices in $V(\Delta)$, lists are presented in Section 4.1.

In lines 4 and 5 of Method 9 we are swapping the values of q and p in case, if we do the disjoint set union on them, p would be merged on q, we want the opposite, q merged on p. Of course this code is not complete as we are missing the cases with (t,q) merging on (t,p), this is an equivalent for loop that can be done inside the one in line 7 or just after that for loop. Another thing to notice is that to handle correctly the merging of the group $(q,q), (p,q), (q,p), (p,p)$ we need to make sure that in the for loop of line 7 the iteration with $t = q$ happens before $t = p$, this can be done with an extra if condition avoiding the case with $t = p$ and doing it at the end of the for loop. One last thing that we have to consider is when we find a loop in line 17, in this case if we are analyzing edge e we will skip the iteration of $\overline{e}$ as we already deleted it in line 16. This happens because we apply the same operations to e and to $\overline{e}$.
One fact about this algorithm is that every time we delete an edge of Γ and we don't insert another one we add at most two pairs of vertices to the list Q. This is because when we remove a self loop we are in fact deleting two edges at the same time. As we don't insert anything to Q when we move the edge without deleting it, we can say the the size of Q increases at most $2 * d$, where d is the number of deleted edges.

Method 9: Performs a Product Stallings folding on vertices p and q

```
Input  : $\Gamma$ – well-labeled product graph
         $v_0$ – basepoint of $\Delta$
         $p$ – vertex to identify
         $q$ – vertex to identify
         $Q$ – queue
         $S$ – list of vertices
Function PRODUCT-FOLD($\Gamma$,$v_0$,$p$,$q$,$Q$,$S$) is
    $p \leftarrow$ FIND-SET($p$)
    $q \leftarrow$ FIND-SET($q$)
    if $p.rank < q.rank$ then
        SWAP($p, q$)
    end
    foreach $t \in S$ do
        $t \leftarrow$ FIND-SET($t$)
        foreach $x \in X^{\pm}$ do
            $Label[x] \leftarrow \emptyset$
        end
        foreach $\mathsf{e} \in (p,t).edges$ do
            $Label[lab(\mathsf{e})] \leftarrow \tau(\mathsf{e})$
        end
        foreach $\mathsf{e} \in (q,t).edges$ do
            DELETE($\mathsf{e}$)
            if $\tau(\mathsf{e}) = (q,t)$ then
                if $Label[lab(\mathsf{e})] = \emptyset$ and $Label[lab(\bar{\mathsf{e}})] = \emptyset$ then
                    INSERT($(p,t)$.edges, $\mathsf{e}$)
                    $Label[lab(\mathsf{e})] \leftarrow (p,t)$
                    $Label[lab(\bar{\mathsf{e}})] \leftarrow (p,t)$
                else
                    if $Label[lab(\mathsf{e})] \neq \emptyset$ then
                        $(k_1,k_2) \leftarrow Label[lab(\mathsf{e})]$
                        PUSH($Q, (p,k_1)$)
                        PUSH($Q, (t,k_2)$)
                    end
                    if $Label[lab(\bar{\mathsf{e}})] \neq \emptyset$ then
                        $(k_1,k_2) \leftarrow Label[lab(\bar{\mathsf{e}})]$
                        PUSH($Q, (p,k_1)$)
                        PUSH($Q, (t,k_2)$)
                    end
                end
            else
                if $Label[lab(e)] = \emptyset$ then
                    INSERT($(p,t)$.edges, $\mathsf{e}$)
                    $Label[lab(\mathsf{e})] \leftarrow \tau(\mathsf{e})$
                else
                    $(k_1,k_2) \leftarrow Label[lab(\mathsf{e})]$
                    PUSH($Q, (p,k_1)$)
                    PUSH($Q, (t,k_2)$)
                end
            end
        end
    end
    if $q = v_0$ then
        $v_0 \leftarrow p$
    end
    DELETE($q$)
    UNION($q, p$)
end
```

Now we can define a new method that will use the Product Stallings foldings. For this method we will consider a graph $\Gamma = \Delta \times \Delta$ where delta is a well-labeled pointed graph, notice that in the Product Stallings foldings we only needed Δ to be labeled and $\Gamma \subseteq \Delta \times \Delta$ with Γ well-labeled. This method is called Product vertex identification, and will get us from Γ to $\Gamma' = \Delta' \times \Delta'$, where Δ' is Δ after identifying two vertices and doing the corresponding Stallings foldings to get a well-labeled graph. The straightforward solution for this problem would be to get Δ from Γ, the diagonal connected component, perform the vertex identification and the foldings in Δ to get Δ', and calculate Γ'. This solution has a time complexity of $O(n^2)$, where $n = E(\Delta) + V(\Delta)$. We can get a more efficient solution using Product Stallings foldings.

The algorithm is really simple, suppose we want to identify vertices $p, q \in V(\Delta)$, we will add (p, q) to an empty queue Q. After that, while the queue Q is not empty, we will take the first element (k_1, k_2) and if they don't represent the same element we will perform a Product Stallings folding on them and add the corresponding new values to Q. When Q is empty, we are done and $\Gamma' = \Delta' \times \Delta'$.

Proposition 5.24. *Let Δ be a finite well-labeled pointed graph and $\Gamma = \Delta \times \Delta$. Let $p, q \in V(\Delta)$. The Product vertex identification of p and q on Γ is finite.*

Proof. Every time we do a Product Stallings folding we reduce the number of vertices of Δ by one. As Δ is finite we only do a finite number of Product Stallings foldings, each of them is finite so the Product vertex identification is finite too. □

Method 10: Product vertex identification of vertices p and q

```
Input : $\Gamma$ – well-labeled product graph
        $v_0$– basepoint of $\Delta$
        $p$ – vertex to identify
        $q$ – vertex to identify
        $S$ – list of vertices
1 Function PRODUCT-IDENTIFY($\Gamma$,$v_0$,$p$,$q$,$S$) is
2     PUSH($Q, (p, q)$)
3     while $Q.next \neq \emptyset$ do
4         $(k_1, k_2) \leftarrow$ POP($Q$)
5         if FIND-SET($k_1$) $\neq$ FIND-SET($k_2$) then
6             PRODUCT-FOLD($\Gamma, v_0, k_1, k_2, Q, S$)
7         end
8     end
9 end
```

To prove the correctness of this method we are going to use the following notation: let $p \in V(\Delta)$, as we will be identifying different vertices using the data structure for disjoint sets, we are going to consider the vertices as equivalence classes. So, if we have $p, q \in V(\Delta)$ holding that in Δ' they represent the same vertex we will say that $[p] = [q]$. In fact $V(\Delta') = \{[p] \mid p \in V(\Delta)\}$. During the Product vertex identification method, we will perform some number of Product Stallings foldings, after each folding the equivalence class will be different. We will denote $[p]_i$ the equivalence class after the i Product Stallings foldings. It is easy to see that, let $p, q \in V(\Delta)$, if $p \neq q$ then $[p]_0 \neq [q]_0$, if $[p]_i = [q]_i$ then $[p]_{i+1} = [q]_{i+1}$, and if the Product vertex identification performs exactly n Product Stallings foldings we have $[p] = [q]$ if and only if $[p]_n = [q]_n$.
With that we can now prove the following proposition:

Proposition 5.25. *Let Δ be a finite well-labeled pointed graph and $\Gamma = \Delta \times \Delta$. Let $p, q \in V(\Delta)$. The Product vertex identification of p and q on Γ has as result $\Gamma' = \Delta' \times \Delta'$, where Δ' is Δ after identifying vertices p and q and doing the corresponding Stallings foldings until the graph is well-labeled again.*

Proof. Let us suppose that the Product vertex identification performs n Product Stallings foldings. Let us denote Γ_i the graph that we have after i Product Stallings foldings and Δ_i the graph Δ after the equivalent i Stallings foldings. We will prove that after each Product Stallings folding, the following holds:
For each pair of edges $e_1, e_2 \in E(\Delta_i)$, not necessarily distinct, such that $lab(e_1) = lab(e_2)$ having $v_1 = \iota(e_1)$, $v_2 = \iota(e_2)$, $v_3 = \tau(e_1)$ and $v_4 = \tau(e_2)$ we have four sequences $\{s^k_0, t^k_0, s^k_1, t^k_1, ..., s^k_{n_k}, t^k_{n_k}\}$, $1 \leq k \leq 4$, with $[s^k_j]_i = [t^k_j]_i$, the unordered pair $(t^k_j, s^k_{j+1}) \in Q$, where Q is the queue used in the method, and $v_k = s^k_0$. Let $u_k = t^k_{n_k}$, we want to see that there is $e \in E(\Gamma_i)$ with $lab(e) = lab(e_1) = lab(e_2)$, $\iota(e) = (u_1, u_2)$ and $\tau(e) = (u_3, u_4)$. We will call these sequences l^k_e, it can be more than one sequence ending at these edge so this is a name for the set of these sequences, not only one.
If this holds we have that at the end, when Q is empty, all $n_k = 0$. So $E(\Gamma') \supseteq E(\Delta' \times \Delta')$. From the algorithm it is easy to see that $E(\Gamma') \subseteq E(\Delta' \times \Delta')$ and $V(\Gamma') = V(\Delta' \times \Delta')$, so we would have $\Gamma' = \Delta' \times \Delta'$.
Let us prove it by induction, when $i = 0$ we have that $\Gamma_0 = \Delta_0 \times \Delta_0$ so is obviously true. Now, suppose that after i iterations it is still holding and Q is not empty. Let us look at the next element of Q, (v_1, v_2). If $[v_1]_i = [v_2]_i$ we don't have to do the Product Stallings folding, because they already represent the same element, and all the sequences that have $(t_j, s_{j+1}) = (v_1, v_2)$, as

unordered pairs, they also hold that $[s_j]_i = [t_{j+1}]_i$ from the definition of the sequences. So we can remove these two elements from these sequences. In the case where $[v_1]_i \neq [v_2]_i$ we perform the Product Stallings folding of these two vertices. As we are identifying these two vertices we will have $[v_1]_{i+1} = [v_2]_{i+1}$, so as before we can remove all the pairs $(t_j, s_{j+1}) = (v_1, v_2)$ from all the lists. Now we will look at each edge $e \in E(\Gamma_i)$ that we have to move or remove. If we move e, we are changing $\iota(e)$ from (v_2, t) to (v_1, t), so if we change the last element of the sequences l_e^1 from v_2 to v_1 the condition is still holding, in case we are changing from (t, v_2) to (t, v_1) it will be l_e^2. If we remove e, it means that there is an edge e' with the same label and $\iota(e) = (v_2, t)$, $\iota(e') = (v_1, t)$, $\tau(e) = (k_1, k_2)$ and $\tau(e') = (k_3, k_4)$. As before we will change the last element of the sequences l_e^1, l_e^2 in the symmetric case, from v_2 to v_1. Now, as we are adding (k_1, k_3) and (k_2, k_4) to Q, we will add two k_3 at the end of the sequences l_e^3 and two k_4 at the end of the sequences l_e^4. Notice that in this second case all the sequences of l_e^k are now in $l_{e'}^k$ after the modifications. $\square$

5.2.2 The algorithm

Now we will improve the time complexity of Proposition 5.23, to do that we will use Product vertex identification. Before doing that, we will need an extra data structure to store information about the connected components of $\Gamma(H) \times \Gamma(H)$. We will use a data structure for disjoint sets, explained in Section 4.3, to keep the set of vertices corresponding to the connected components. We can initialize this data structure unifying all the endpoints of each edge.
Another thing that we will need is some indicator that tells us if each component is a tree (Definition 3.22) or not. To do that, we will have a table that we will call $cycle : V(\Gamma(H) \times \Gamma(H)) \to \mathbb{Z}$. Initially all vertices $p \in V(\Gamma(H) \times \Gamma(H))$ will have $cycle(p) = 0$. We will only care about the value of $cycle$ of the elements in $V(\Gamma(H) \times \Gamma(H))$ that are representative elements of some set (connected component). For that reason every time we merge two connected components into one we will update the value of $cycle$ for the representative of the new component. Let us say we have a connected component $C \subseteq \Gamma(H) \times \Gamma(H)$, with representative $C_r \in V(\Gamma(H) \times \Gamma(H))$, then $cycle(C_r) = |E^+(C)| - |V(C)| + 1$, where $E^+(C)$ are the edges e of the component C with $lab(e) \in X^+$. We can see that $cycle(C_r) \geq 0$ and $cycle(C_r) = 0$ if and only if C is a tree. So now we have an easy way to check if a component is a tree.
Once we have $\Gamma(H) \times \Gamma(H)$, the first thing we will do is to update the disjoint set data structure for $V(\Gamma(H) \times \Gamma(H))$ to store the connected components and calculate the corresponding value of $cycle$ for each component. This can be done in time $O(n_e + n_v \cdot log^*(n_v))$ where $n_e = |E(\Gamma(H) \times \Gamma(H))|$ and $n_v = |V(\Gamma(H) \times \Gamma(H))|$. As we want to find the components that are trees, we will have a linked list T with the representatives of the connected components that have $cycle(C_r) = 0$.

During each Product Stallings folding the connected components and the values of $cycle$ of some components will change, now we will describe how to update them efficiently and how to keep T updated.
Every time we identify the vertices (p, t) and (q, t) we have two cases: if they are already at the same component we are in fact removing a vertex, so if they are at component C we do $cycle(C_r) \leftarrow cycle(C_r) + 1$. If they are from different components C^p and C^q, we will merge them and get the component C with $cycle(C_r) \leftarrow cycle(C_r^p) + cycle(C_r^q)$. Now, when we are moving the edges from (q, t) to (p, t), every time we remove an edge and we don't add it again we will do $cycle(C_r) \leftarrow cycle(C_r) - 1$. We can see that the time complexity of the Product Stallings folding is $O(P_c + n \cdot log^*(n))$, where P_c is the original complexity and $n = |\Gamma(H)|$, and we keep $cycle$ and the connected components updated.
To keep T updated we have to notice that, although after each Product vertex identification the graph will be a product graph and for each connected component C we will have $cycle(C_r) \geq 0$, we might have cases with $cycle(C_r) < 0$ during the Product Stallings foldings. This is because until the end of the Product vertex identification the graph is not the whole product graph, as we saw in the proof of Proposition 5.25, so the sets of the disjoint set data structure might not represent connected components in the intermediate steps of the Product vertex identification. So, we will still keep in T only the components (or sets) with $cycle(C_r) = 0$, to do that before identifying (p, t) and (q, t) we will do the following: if $cycle(C_r^p) = 0$ we will do DELETE(C_r^p)), and if $C^p \neq C^q$ we will do the same for C_r^q. Now, after updating the value of $cycle(C_r)$, if $cycle(C_r) = 0$ we will do

INSERT(T, C_r). This way we will keep T updated without changing the original time complexity of the Product Stallings folding.

With these extra data structures we can now prove the final complexity of our algorithm, but first we need to prove the complexity of the Product vertex identifications.

Proposition 5.26. *Let Δ be a connected well-labeled pointed graph and let $\Gamma = \Delta \times \Delta$. We can do all valid sequences of Product vertex identifications in Γ with time complexity $O(n^2)$, where $n = |E(\Delta)|$.*

Proof. First, it is easy to see that as Δ is connected $|V(\Delta)|$ is $O(n)$. Let $\Gamma = \Gamma_0, \Gamma_1, \Gamma_2, ..., \Gamma_k = \Gamma'$ be the sequence of graphs that we get after each Product Stallings folding, and let $\Gamma_i \subseteq \Delta_i \times \Delta_i$. It is true that $|V(\Delta_i)| = |V(\Delta_{i+1})| + 1$, so we will do $O(n)$ Product Stallings foldings. As $|X^{\pm}|$ is constant, each Product Stallings folding has amortized complexity $O(n + log^*(n))$, because we identify $O(n)$ pairs of vertices of Γ, and we unify one pair of vertices of Δ. So in the end, we have indeed time complexity $O(n^2)$. $\square$

Now, we can prove the final theorem.

Theorem 5.27. *Let $H \leq_{fg} F(X)$ and $H = \langle h_1, h_2, ..., h_k \rangle$ with $n = \sum_{i=1}^{k} |h_i|$. We can compute H_{mal} in $O(n^2 \cdot log^*(n))$.*

Proof. As we did in the proof of Proposition 5.23, we will prove that we can get a sequence $H = H_0 \leq H_1 \leq ... \leq H_k = H_{mal}$ with $k \leq n$, such that we can get from $\Gamma(H_i) \times \Gamma(H_i)$ to $\Gamma(H_{i+1}) \times \Gamma(H_{i+1})$ using one Product vertex identification. The first thing we have to do is calculate $\Gamma(H) \times \Gamma(H)$ and initialize the values of T and *cycle*, this can be done in $O(n^2 \cdot log^*(n))$. Now, suppose that we have $\Gamma(H_i) \times \Gamma(H_i)$ with T and *cycle* updated, if there is an element $(p, q) \in T$ with $[p]_i \neq [q]_i$, it means that we have a connected component, that is not the diagonal, with a cycle. From Proposition 5.20 we know that if u is the element of $F(X)$ represented by the label of a reduced path from v_0 to p in $\Gamma(H_i)$ and v the same for q, then $H_i \cap gH_ig^{-1} \neq 1$, where $g = uv^{-1}$. So $\langle g, H_i \rangle \leq H_{mal}$. Let $H_{i+1} = \langle g, H_i \rangle$, we can get $\Gamma(H_{i+1}) \times \Gamma(H_{i+1})$ from doing a Product vertex identification of p and q on $\Gamma(H_i) \times \Gamma(H_i)$. From Proposition 5.26 and using the fact that if we update T and *cycle* then the total complexity gets multiplied by $log^*(n)$, we have that the total complexity is $O(n^2 \cdot log^*(n))$. $\square$

6 Finitary free product of cyclic and finite groups

In this section we will be working with subgroups of a finitary free product of cyclic groups and finite groups. We are going to develop a way to represent these subgroups graphically using labeled graphs similar to the ones used for the free group. Once we have that, we are going to see how can we use these graphs for determining if a subgroup is malnormal. Like in the previous section, we will describe an algorithm to calculate the malnormal closure of a subgroup and then we will improve it using Product Stallings foldings (Section 5.2.1).

First we will define the free product and some theorems about it that will help us build the Extended Stallings graph.

Definition 6.1 (Free product)**.** *Let A and B be groups with presentations $A = \langle a_1, \ldots ; r_1, \ldots \rangle$ and $B = \langle b_1, \ldots ; s_1, \ldots \rangle$ respectively, where the sets of generators $\{a_1, \ldots\}$ and $\{b_1, \ldots\}$ are disjoint. The free product $A * B$ of groups A and B is the group $A * B = \langle a_1, \ldots, b_1, \ldots ; r_1, \ldots, s_1, \ldots \rangle$.*

Now we will define normal forms, a way to represent the elements of this type of groups.

Definition 6.2 (Normal form)**.** *A reduced sequence or normal form is a sequence $g_1, \ldots, g_n, n \geq 0$, of elements of $A * B$ such that each $g_i \neq 1$, each g_i is in one of the factors A or B and successive g_i and g_{i+1} are not in the same factor.*

We can now prove that normal forms are a good way to identify the elements of $A * B$.

Proposition 6.3. *Consider the free product $A * B$. Then the following two equivalent statements hold.*

- *If $w = g_1g_2 \dots g_n, n > 0$, where $g_1, \dots, g_n$ is a normal form, then $w \neq 1$ in $A * B$.*
- *Each element $w \in A*B$ can be uniquely expressed as a product $w = g_1 \dots g_n$ where $g_1, \dots, g_n$ is a normal form.*

Proof. First of all we will show that both statements are equivalent. In the second statement, it is understood that 1 is the product of the empty sequence. Hence, the second statement implying the first one is immediate. Assume the first statement holds. Let $w = g_1 \dots g_n$ and $w = h_1 \dots h_m$ be reduced. Then $1 = g_1 \dots g_n h_m^{-1} \dots h_1^{-1}$. In order for the sequence $g_1, \dots, g_n, h_m^{-1}, \dots, h_1^{-1}$ not to be reduced, it is necessary that h_m be in the same factor as g_n and $1 = g_n h_m^{-1}$, so $g_n = h_m$. As induction argument $n = m$ and $g_i = h_i$. So both statements are equivalent.
We prove the proposition by using a homomorphism into a permutation group. Let W be the set of all reduced sequences from $A * B$. For each element $a \in A$, define a permutation $\overline{a}$ of W as follows. If $a = 1$, $\overline{a}$ is the identity. If $a \neq 1$ and $(g_1, \dots, g_n)$ is a reduced sequence then:

$$\overline{a}((g_1, \dots, g_n)) = \begin{cases} (a, g_1, \dots, g_n) & \text{if } g_1 \in B \\ (ag_1, \dots, g_n) & \text{if } g_1 \in A, ag_1 \neq 1 \\ (g_2, \dots, g_n) & \text{if } g_1 = a^{-1} \end{cases}$$

To verify that $\overline{a}$ is a permutation of W we note that $\overline{a^{-1}}$ is the inverse of $\overline{a}$. An easy check shows that if $a, a' \in A$ then $\overline{aa'} = \overline{a}\overline{a'}$. The map $\phi : a \to \overline{a}$ is thus a homomorphism of A into $S(W)$, the group of permutations of W. Define a homomorphism $\psi : b \to \overline{b}$ similarly. We thus have a homomorphism $\phi * \psi : A * B \to S(W)$. Now any element w of $A * B$ can certainly be written as some product $w = g_1 \dots g_n$ where $g_1, \dots, g_n$ is reduced. Note that the permutation $\phi * \psi(w)$ sends the empty sequence to the sequence $(g_1, \dots, g_n)$. Thus $w \neq 1$ if $n > 0$. □

We will be working with subgroups of groups of the form $G_1 * G_2 * \dots * G_n$. Where G_i are finite groups or $\mathbb{Z}$. We can see that $F_m \cong \underbrace{\mathbb{Z} * \mathbb{Z} * \dots * \mathbb{Z}}_{m \text{ times}}$, so we can also describe the groups with $G_1 * G_2 * \dots * G_n * F_m$. Where G_i are finite groups, and F_m is a free group with a basis of m elements.

6.1 Extended Stallings graphs

In this section we will present a different type of labeled graphs that will allow us to represent finitely generated subgroups of groups of the form $G_1 * G_2 * \dots * G_n$ geometrically. The idea of this type of graphs is inspired in the work of Markus-Epstein with geometric representation of subgroups of amalgamations of two finite groups in [6]. As with the Stallings graph, we want a bijection between these type of graphs and the subgroups of $G_1 * G_2 * \dots * G_n$. To do that we will need to prove equivalent propositions to Propositions 5.14, 5.15 and 5.16.
To do that, instead of working with freely reduced paths we will be working with normally reduced paths.
From now on, when we define a group $G = G_1 * \dots * G_n$ and we let $X = X_1 \cup \dots \cup X_n$ be a generator set, we are assuming that each X_i generates G_i and that they are disjoint.

Definition 6.4 (Normally reduced word)**.** *Let $G = G_1 * \dots * G_n$ be a group and let $X = X_1 \cup \dots \cup X_n$ be the set of generators of this group. We call a word $w \in (X^{\pm})^*$, $w = a_1 \dots a_m$ with $a_i \in X^{\pm}$, normally reduced if there is no nonempty subword $w' = a_{j_1} a_{j_1+1} \dots a_{j_2-1} a_{j_2}$ with $a_i \in X_k^{\pm}$ for $j_1 \leq i \leq j_2$ satisfying $w' =_G 1$.*

Notice that the definition of normally reduced words is stronger than the one for freely reduced words. Also, in the free group these two definitions are equivalent as we don't have any relation. Now we can define a normally reduced path in the obvious way:

Definition 6.5 (Normally reduced path)**.** *Let $G_1 * \dots * G_n$ be a group and let $X = X_1 \cup \dots \cup X_n$ be the set of generators of this group. Let Γ be a labeled graph, with labels on $X^{\pm}$. We call a path p in Γ normally reduced if the word $lab(p)$ is normally reduced.*

From now on, when we are talking about labeled graphs, we will be talking about graphs with labels on $X^{\pm}$, where $X = X_1 \cup \ldots \cup X_n$ is the generator set of the group $G_1 * \ldots * G_n$, where G_i are finite groups or $\mathbb{Z}$.
Now we will define a concept equivalent to core graph for normally reduced paths.

Definition 6.6 (Normal core graph). *Let Γ be a labeled graph, the normal core of Γ at $v \in V(\Gamma)$ is the subgraph $NCore(\Gamma, v)$ induced by the union of the normally reduced closed paths in Γ closed at v.*

As we showed in Lemma 3.27 for the core graph, we can do the same for normal core graphs and the proof is equivalent.

Lemma 6.7. *Let Γ be a labeled graph. Let $\Gamma' = NCore(\Gamma, v)$. Then:*

1. *Γ' is connected and contains the vertex v.*
2. *Γ' has no degree-one vertices, except possibly for the vertex v.*
3. *$Lab(\Gamma', v) = Lab(\Gamma, v)$.*

Definition 6.8 (Complete graph). *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let (Γ, v) be a labeled pointed graph. Let $Lab(\Gamma, v) = H \leq G$. We call this graph complete if for every normally reduced word w representing an element of H, there is a path p closed at v in Γ with $lab(p) = w$.*

Fact 6.9. *Let (Γ, v) be a complete graph, then $NCore(\Gamma, v)$ is also complete.*

We can now start proving the equivalent propositions for the existence and uniqueness of the Extended Stallings graphs. First, as normally reduced is stronger than freely reduced, we have that from Lemma 5.13 follows the following lemma with the same proof.

Lemma 6.10. *Let (Γ, v) be a complete normal core pointed well-labeled graph with $Lab(\Gamma, v) = H$. For every prefix w of a normally reduced word representing an element $h \in H$ there is a unique path p in Γ with $\iota(p) = v$, and $lab(p) = w$.*

Let us define some useful concepts to represent groups and cosets of a subgroup graphically. The first concept is the Cayley graph of a group, first presented by A. Cayley in [2].

Definition 6.11 (Cayley graph). *Suppose that G is a group and X is a generating set of G. The Cayley graph $Cayley(G)$ is a well-labeled graph constructed as follows:*

- *Each element $g \in G$ is assigned a vertex: the vertex set $V(Cayley(G))$ is identified with G.*
- *The edges are labeled with the set $X^{\pm}$.*
- *For any $g \in G$ and $x \in X^{\pm}$, the vertices corresponding to the elements g and gx are connected by and edge e with $lab(e) = x$, initial vertex g and terminal vertex gx.*

Now let us define a similar concept for cosets of a subgroup on a group. It was first presented by O. Schreier in [7].

Definition 6.12 (Schreier graph). *Suppose that G is a group, X is a generating set of G and $H \leq G$. The Schreier graph $Schreier(G, H)$ is a well-labeled graph constructed as follows:*

- *Each coset Hg, with $g \in G$, is assigned a vertex: the right cosets of H in G are identified with $V(Schreier(G, H))$.*
- *The edges are labeled with the set $X^{\pm}$.*
- *For any $g \in G$ and $x \in X^{\pm}$, the vertices corresponding to the elements Hg and Hgx are connected by and edge e with $lab(e) = x$, initial vertex Hg and terminal vertex Hgx.*

We can see that $Cayley(G) = Schreier(G, 1)$. Also, we can consider $Cayley(G)$ and $Schreier(G, H)$ as pointed graphs, with basepoints corresponding to the elements 1 and H respectively.

Convention 6.13. *When we apply a morphism of labeled graphs ϕ to the Cayley graph of some group G, $Cayley(G)$. The image of this morphism has to be also a well-labeled graph. So it will always be the Schreier graph of the cosets of some $H \leq G$.*

Lemma 6.14. *Given a complete normal core well-labeled graph Γ. If $v \in V(\Gamma)$ and there is an edge $e \in E(\Gamma)$ with $\iota(e) = v$ and $lab(e) \in X_i^{\pm}$, where G_i is finite, then for every word $w \in (X_i^{\pm})^*$ with $w =_{G_i} 1$ there is a path p in Γ closed at v with $lab(p) = w$.*

Proof. As Γ is a complete normal core graph, for every edge there is a normally reduced closed path at the basepoint passing trough it. Let wgw' be a normally reduced word representing an element of $Lab(\Gamma)$, with $g \in (X_i^{\pm})^+$ and the last element of w and the first of w' not in $X_i^{\pm}$. Let us look at the Cayley graph of G_i. $Cayley(G_i)$ is a well-labeled pointed core graph. If we take the path p in $Cayley(G_i)$, with $\iota(p) = v_0$, the basepoint, and $lab(p) = g$, we get the vertex $v_1 = \tau(p)$. Now, for all the normally reduced paths p' in $Cayley(G_i)$, paths without cycles, with $\iota(p') = v_0$ and $\tau(p') = v_1$ we have that $wlab(p')w'$ is a normally reduced word representing the same element as wgw', so there is a closed path with this label on Γ. From that we can get that all the maximal connected subgraphs of Γ with edges labeled only in $X_i^{\pm}$ are isomorphic to $\phi(Cayley(G_i))$ where ϕ is a morphism of well-labeled graphs.

The morphism ϕ is defined by comparing closed paths $v \xrightarrow{u'} q' \xrightarrow{u''} v$ and $v_0 \xrightarrow{u'} q \xrightarrow{u''} v_0$ in $Cayley(G_i)$ and Γ, respectively, making $\phi(q) = q'$. Here v is the terminal vertex of the path starting at the basepoint in Γ with label w. We can see that ϕ is a well-defined morphism since $x =_{G_i} 1$ implies that there is a path in Γ closed at v with label x. Note that this morphism is different for each maximal connected component with edge labels in $X_i^{\pm}$ of Γ. □

Proposition 6.15. *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let $K \leq H \leq G$. Suppose (Γ_1, v_1) and (Γ_2, v_2) are complete normal core well-labeled graphs such that $Lab(\Gamma_1, v_1) = K$ and $Lab(\Gamma_2, v_2) = H$. Then there exists a unique morphism of pointed labeled graphs $\phi : \Gamma_1 \to \Gamma_2$.*

Proof. The uniqueness follows from Lemma 3.33. We have to prove that such π exists.
As Γ_1 is a normal core graph, every vertex in $V(\Gamma_1)$ is part of a normally reduced path. Let w be the prefix of a normally reduced word representing an element $k \in K$. From Lemma 6.10 we have that there is a path p_1 in Γ_1 with $\iota(p_1) = v_1$ and $lab(p_1) = w$, and also a path p_2 in Γ_2 with $\iota(p_2) = v_2$ and $lab(p_2) = w$.
We define $\phi(\tau(p_1)) = \tau(p_2)$, and the image of the edges as the only possible edge following the morphism conditions, it is unique because the graphs are well-labeled. We need to check that this is well defined.
For each vertex of Γ_1 we will consider w as the shortest word holding the conditions and we will be doing induction over the length of the word. The base case is the basepoint which is obviously true.
Suppose we take another prefix $w' \neq w$ of a reduced word representing an element of K such that the path p_1' in Γ_1 with $\iota(p_1') = v_1$ and $lab(p_1') = \underline{w'}$ has $\tau(p_1') = \tau(p_1)$. We will see that the equivalent path p_2' in Γ_2 has $\tau(p_2') = \tau(p_2)$. Let $p = p_1\overline{p_1'}$, this is a closed path but maybe it is not normally reduced. If it is not, there is a subpath p' such that $p = tp'\overline{t'}$, where t is a strict prefix of p_1 and t' a strict prefix of p_1', that has all the edges at some $X_i^{\pm}$ and with $lab(p') =_G 1$. If $G_i \cong \mathbb{Z}$ then we can see that the last edge of p_1 is the same as the last edge of p_1', so if u is the initial vertex of this edge we have that if $\phi(u)$ is well defined then $\phi(\tau(p_1))$ is well defined, that holds for the induction assumption. Now if G_i is finite, from Lemma 6.14 we have that p' is a closed path, so $\phi(\tau(p_1))$ if $\phi(\iota(p'))$ is well defined, that holds for the induction assumption. In the case p is normally reduced, it represents an element of $K \leq H$ so there is also a closed path with label $lab(p)$ in Γ_2, hence $\phi(\tau(p_1)) = \phi(\tau(p_1'))$. □

Proposition 6.16. *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let $H \leq_{fg} G$. There is a complete normal core well-labeled pointed graph Γ, such that $Lab(\Gamma) = H$.*

Proof. Let us consider the labeled graph Δ where we have a bijection f between right cosets of H in G and the elements of $V(\Delta)$. The edge e with $\iota(e) = v_1$, $\tau(e) = v_2$ and $lab(e) = x$ with

$v_1, v_2 \in V(\Delta)$ and $x \in X^{\pm}$ is in $E(\Delta)$ if and only if $f^{-1}(v_1)x = f^{-1}(v_2)$, where $f^{-1}(v_i)$ is the right coset assigned to each vertex.
This graph is connected. Let us call $v_0 = f(H) \in V(\Delta)$, for each vertex $v_1 = f(Hg)$, with $g \in G$, there is a path p with $lab(p) =_G g$ such that $\iota(p) = v_0$ and $\tau(p) = v_1$.
The graph is complete. From the construction fo the graph, for each vertex $v \in V(\Delta)$ and each $x \in X^{\pm}$, there is an edge $e \in E(\Delta)$ with $\iota(e) = v$ and $lab(e) = x$. So for each normally reduced word w there is a path p with $\iota(p) = v_0$ and $lab(p) = w$. If w represents an element $h \in H$, we have that $\tau(p) = f(Hh) = f(H) = v_0$.
Δ is well-labeled. Suppose we have $e_1, e_2 \in E(\Delta)$ with $lab(e_1) = lab(e_2) = x$ and $\iota(e_1) = \iota(e_2) = v$, and $u_i = \tau(e_i)$. Then we have $f^{-1}(u_1) = f^{-1}(v)x = f^{-1}(u_2)$, so $u_1 = u_2$ and hence $e_1 = e_2$.
It is obvious from the definition of the graph that $Lab(\Delta, v_0) = H$
If we define $\Gamma = HCore(\Delta, v_0)$, from Lemma 6.7 we have that this graph holds all the conditions. □

Proposition 6.17. *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let $H \leq_{fg} G$. Suppose (Γ_1, v_1) and (Γ_2, v_2) are complete normal core pointed well-labeled graphs and $Lab(\Gamma_1, v_1) = H = Lab(\Gamma_2, v_2)$. Then there exists a unique isomorphism of pointed labeled graphs $\pi : \Gamma_1 \to \Gamma_2$.*

Proof. The uniqueness of π comes from Lemma 3.33.
Since $H \leq H$. by Proposition 6.15 there is a unique morphism of pointed graphs $\pi : (\Gamma_1, v_1) \to (\Gamma_2, v_2)$. We claim that π is an isomorphism of pointed graphs. Proposition 6.15 also implies there is a unique morphism of pointed graphs $\pi' : (\Gamma_2, v_2) \to (\Gamma_1, v_1)$. Therefore $\pi' \circ \pi : (\Gamma_1, v_1) \to (\Gamma_1, v_1)$ is a morphism, from Lemma 3.33 it is unique, namely the identity map on (Γ_1, v_1). We can use a symmetric argument to show that $\pi \circ \pi'$ is the identity map on (Γ_2, v_2). Therefore π is an isomorphism. □

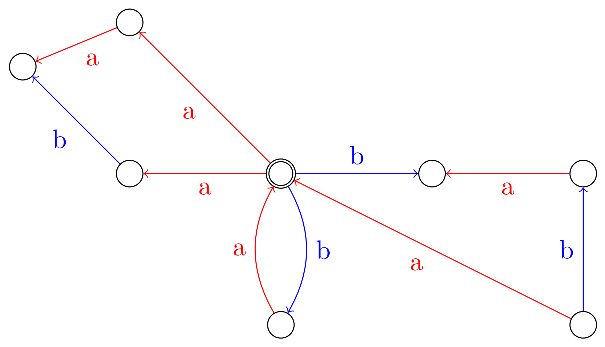
Now that we know the existence and uniqueness of the Extended Stallings graph of a subgroup H we can define:

Definition 6.18. *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let $H \leq_{fg} G$. We call the unique complete normal core pointed well-labeled graph (Δ, v_0) such that $Lab(\Delta, v_0) = H$, the Extended Stallings graph of the subgroup H, and we denote it by $\Gamma(H)$.*

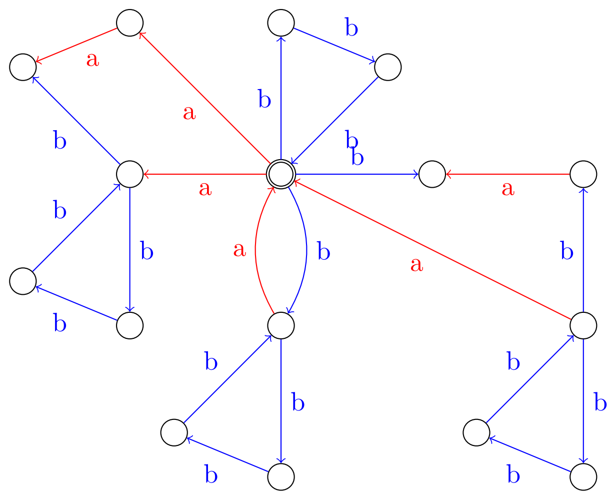
Notice that we gave the same name to the extended graph as the Stallings graph for free groups. This is because free groups are also of the form $G_1 * \ldots * G_n$ and the Extended Stallings graph is the same as the Stallings graph for them.

Theorem 6.19. *Let $\{h_1, h_2, \ldots, h_m\}$ be a generator set for $H \leq G_1 * \ldots * G_n$, where G_i are finite or cyclic groups. There is an algorithm that builds $\Gamma(H)$ in $O(n^2)$, where $n = \sum_{i=1}^{m} |h_i|$.*

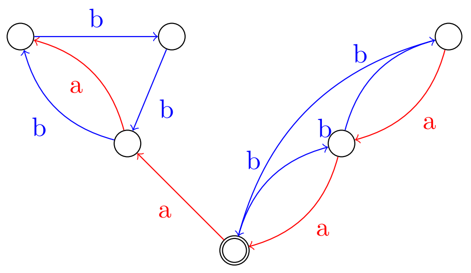
Proof. As with the standard Stallings algorithm, we build the flower graph $\mathcal{F}(H)$. We know that $Lab(\mathcal{F}(H)) = H$. Now we will create the extended flower graph $\mathcal{E}(H)$ from $\mathcal{F}(H)$. For each $e \in E(\mathcal{F}(H))$ with $lab(e) \in X_i$, where G_i is finite, we will attach one copy of the graph $Cayley(G_i)$ at $\iota(e)$ by merging $\iota(e)$ to some vertex of $Cayley(G_i)$. Still $Lab(\mathcal{E}(H)) = H$.
If we now do the standard folding process until we get a well-labeled pointed graph $G(H)$, we know that $Lab(G(H)) = H$, and we can easily see from the construction of $\mathcal{E}(H)$ that $G(H)$ is a complete normal core graph. So $G(H) = \Gamma(H)$. □



(a) The flower graph $\mathcal{F}(H)$ where $H = \langle aba^{-2}, ba^{-1}b^{-1}a, ba\rangle$



(b) We attach the Cayley graph of $\langle b \mid b^3\rangle$ at the vertices with an outgoing edge with label b



(c) The graph after folding the necessary edges. This graph is $\Gamma(H)$.

Figure 11: The folding process from the flower graph to the Extended Stallings graph of a subgroup of $G = \langle a \mid\rangle * \langle b \mid b^3\rangle$.

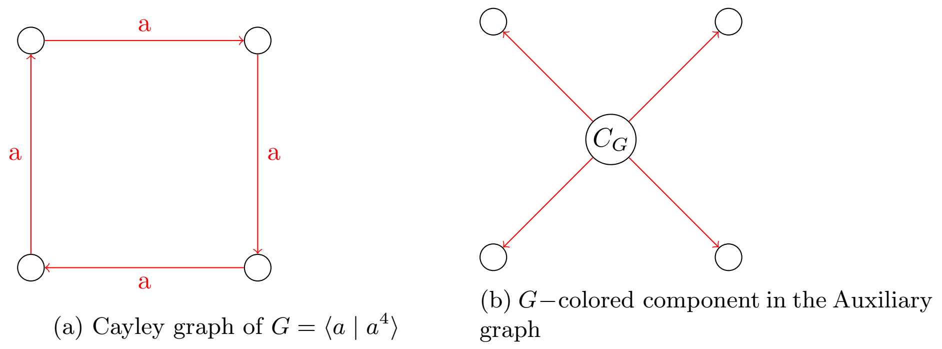
6.2 Malnormal closure algorithm

As with the free group, first we are going to see some relations between a subgroup H being malnormal and some properties of $\Gamma(H)$. Right after that we will be able to describe an algorithm to calculate the malnormal closure of H. Using Product Stallings foldings (Section 5.2.1) we will improve the time complexity of the algorithm finding our final result.
Before starting with the properties we need to define some extra concepts.

Definition 6.20 (Colored component)**.** *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let Γ be a labeled graph. We call a colored component of Γ, a maximal connected component where all the edges are labeled with generators of the same factor G_i, $X_i^{\pm}$. We will call this a G_i-colored component.*

Notice that given a labeled graph Γ, each $v \in V(\Gamma)$ can be part of more than one colored component but at most one G_i-colored component for each i.

Definition 6.21 (Auxiliary graph)**.** *Let $G = G_1 * \ldots * G_n$ be a group and let $X = X_1 \cup \ldots \cup X_n$ be the set of generators of this group. Let Γ be a labeled graph. We define the Auxiliary graph of Γ, $A(\Gamma)$, as the not labeled graph obtained from Γ by the following process: for each G_i-colored component of Γ, where G_i is a finite group, we remove all the edges of the component and add one extra vertex connected to all the previous vertices of the G_i-colored component with edges with no label. At the end we remove the labels of the edges from infinite groups.*



(a) Cayley graph of $G = \langle a \mid a^4 \rangle$

(b) $G-$colored component in the Auxiliary graph

Figure 12: The two $G-$colored component are equivalent, one in the labeled graph and the second one in the Auxiliary graph

Notice that the auxiliary graph is not labeled, all the edges we add have no label, and we remove the label from the edges that remain the same.

All the connected components in Γ are connected components in $A(\Gamma)$, but maybe with different set of edges and new vertices. It is easy to see that we can compute $A(\Gamma)$ from Γ in $O(|E(\Gamma)|+|V(\Gamma)|)$. We can now start with the malnormality conditions.

Convention 6.22. *When we say that a labeled graph Γ not pointed is complete, it means that (Γ, v) is complete for each $v \in V(\Gamma)$.*

Lemma 6.23. *Let (Γ, v_0) be a complete normal core pointed well-labeled graph, all the connected components of $\Gamma \times \Gamma$ are complete well-labeled graphs with all the G_i-connected components, G_i finite, isomorphic to the image of some morphism applied to $Cayley(G_i)$.*

Proof. As we saw in Proposition 6.17, (Γ, v_0) is the only complete normal core pointed well-labeled graph with $Lab(\Gamma, v_0) = H$ up to isomorphism. So, from the construction of this graph in Theorem 6.19 we know that all the G_i-colored components with G_i finite are isomorphic to the image of some morphism applied to $Cayley(G_i)$.
It is easy to see that all the G_i-colored components with G_i finite in $\Gamma \times \Gamma$ are also isomorphic to the image of some morphism applied to $Cayley(G_i)$. This is a sufficient condition for a well-labeled graph to be complete. □

Lemma 6.24. *Let Γ be a complete connected well-labeled graph where all the G_i-connected components, with G_i finite, are isomorphic to the image of some morphism applied to $Cayley(G_i)$. We have $Lab(\Gamma, v) = 1$ for each $v \in V(\Gamma)$ if and only if $A(\Gamma)$ is a tree and each G_i-connected component, with G_i finite, is isomorphic to $Cayley(G_i)$. If for some $v_0 \in V(\Gamma)$ we have $Lab(\Gamma, v_0) \neq 1$, then $Lab(\Gamma, v) \neq 1$ for all $v \in V(\Gamma)$.*

Proof. First, let us see that we have $Lab(\Gamma, v) = 1$ for each $v \in V(\Gamma)$ or $Lab(\Gamma, v) \neq 1$ for each $v \in V(\Gamma)$. If there is some vertex $v_0 \in V(\Gamma)$ with $Lab(\Gamma, v_0) \neq 1$ it means there is a path p closed at v_0 with $lab(p) \neq_G 1$. As Γ is connected, for all $v \in V(\Gamma)$, let p_v be a path from v to v_0, we have that $lab(p_v p \overline{p_v}) \neq_G 1$ so $Lab(\Gamma, v) \neq 1$.
Now, let us see that there is one closed normally reduced path p with $1 \neq lab(p) \in G_i$ with G_i finite

if and only if the G_i-colored component containing p is not isomorphic to $Cayley(G_i)$. First of all, as p is normally reduced and $lab(p) \in G_i$, from Proposition 6.3 we have that p is fully contained in a G_i-colored component. It cannot be that the colored component is isomorphic to $Cayley(G_i)$, because then $lab(p) = 1$. If there is a G_i-colored component with G_i finite that is not isomorphic to $Cayley(G_i)$, then there is a closed path in it with $1 \neq lab(p) \in G_i$. All this follow from the Convention 6.13 about the morphisms applied to $Cayley(G_i)$.
There is a closed normally reduced path p such that $lab(p)$ is not fully contained in one finite factor if and only if $A(\Gamma)$ has a closed reduced cycle. Let us start with the case where p is a closed normally reduced path, we can divide the path in $p = p_1p_2, ..., p_n$ where each p_i is a maximal subpath in a G_i-colored component. We will get a closed reduced path in $A(\Gamma)$ $p' = p'_1p'_2, \ldots, p'_n$ as follows. If p_i is in an infinite group colored component, then $p'_i = p_i$. If it is from a finite group, let c_{G_i} be the central extra vertex of the colored component in $A(\Gamma)$, we define $p'_i = \{\iota(p_i), e_1, c_{G_i}, e_2, \tau(p_i)\}$, where e_1, e_2 are the corresponding edges from the center to the endpoints of p_i. So we can see that p' is a valid closed reduced path in $A(\Gamma)$. If we want to go from a closed reduced path in $A(\Gamma)$ to a normally reduced closed path in Γ we just do the reversed transformations. □

Proposition 6.25. *Let $G = G_1 * \ldots * G_n$ and $H \leq_{fg} G$. H is malnormal in G if and only if the connected components of $A(\Gamma(H) \times \Gamma(H))$, that are not the diagonal, are trees and each G_i-connected component of $\Gamma(H) \times \Gamma(H)$ not in the diagonal component, with G_i finite, is isomorphic to $Cayley(G_i)$.*

Proof. From Lemma 6.23 we know that $\Gamma(H) \times \Gamma(H)$ is a complete well-labeled graph where all the G_i-connected components, G_i finite, are isomorphic to the image of some morphism applied to $Cayley(G_i)$.
Suppose that we have a connected component C of $\Gamma(H) \times \Gamma(H)$ such that $A(C)$ has a reduced cycle or there is some G_i-connected component in C not isomorphic to $Cayley(G_i)$, G_i finite. There is a path p in C closed at (v_1, v_2) with $lab(p) \neq_G 1$. That means that at $\Gamma(H)$ we have the same closed path p, closed at v_1 and at v_2. Let $g =_G lab(p)$, and u_i be the label of a path from v_0 to v_i in $\Gamma(H)$. Let $g_i =_G u_i$, we have $g \in g_1^{-1}Hg_1 \cap g_2^{-1}Hg_2$. From here we get $g_2gg_2^{-1} \in H \cap g_2g_1^{-1}H(g_2g_1^{-1})^{-1}$. As $g \neq 1$ we have so $g_2gg_2^{-1} \neq 1$, and also $g_2g_1^{-1} \notin H$ because that would mean $v_1 = v_2$ and (v_1, v_2) is not in the diagonal component so H is not malnormal.
Now suppose H is not malnormal. Then we have $g \in G \setminus H$ such that $H \cap gHg^{-1} \neq 1$. Let w be a normally reduced word representing g and $w = u\overline{v}$, where v is the longest prefix of $\overline{w}$ that we can get as a label of a path at $\Gamma(H)$ starting at v_0, ending at v_1. Then every element from gHg^{-1} has a normally reduced path p, with a label representing this element, in $\Gamma(gHg^{-1})$ closed at the basepoint, with $lab(p) = ut\overline{u}$, where t is a normally reduced word. We have that $H \cap gHg^{-1} \neq 1$ so there is a normally reduced word $h \neq 1$ with $uh\overline{u}$ representing an element x such that $x \in H \cap gHg^{-1}$ and $uh\overline{u}$ is normally reduced. So we have a path at $\Gamma(H)$ from v_0 to v_2 with label u. As we have a normally reduced closed path with the same nonempty label closed at v_1 and v_2. We have that $Loop(\Gamma(H), v_1) \cap Loop(\Gamma(H), v_2) \neq \emptyset$, and so $Loop(\Gamma(H) \times \Gamma(H), (v_1, v_2)) \neq \emptyset \implies Lab(\Gamma(H) \times \Gamma(H), (v_1, v_2)) \neq 1$. So from Lemma 6.24 we have that there is at least one connected component C in $\Gamma(H) \times \Gamma(H)$ that is not the diagonal with $A(C)$ not a tree or some G_i-connected component, with G_i finite, not isomorphic to $Cayley(G_i)$. □

Corollary 6.26. *Let $G = G_1 * \ldots * G_m$ and $H \leq_{fg} G$. Given $\Gamma(H)$ there is an algorithm that finds some $g \in G \setminus H$ such that $gHg^{-1} \cap H \neq 1$ or determines that H is a malnormal subgroup of G in $O(n^2)$.*

Proof. As seen in Theorem 6.19 and Proposition 3.38 we can compute $\Gamma(H)$ and $\Gamma(H) \times \Gamma(H)$ in $O(n^2)$, we also can compute $A(\Gamma(H) \times \Gamma(H))$ in $O(n^2)$. We can check for cycles in the connected components that are not the diagonal in $A(\Gamma(H) \times \Gamma(H))$, if we find a cycle component, we take a vertex (v_1, v_2) from it that is not one of the extra vertices of the Auxiliary graph. This can be done in $O(n^2)$. If we don't found any, we check all the G_i-colored components of finite factors not in the diagonal. If the number of vertices in the component is equal to $|G_i|$, it is isomorphic to $Cayley(G_i)$, if the number is less we take one vertex (v_1, v_2) from it. This can also be done in $O(n^2)$, because every vertex is at most at m colored components, where m is constant.
If we don't found any vertex (v_1, v_2) then H is malnormal. Let u_1 and u_2 be the words from

the first part of the proof of Proposition 6.25, we can get them in $O(n^2)$ time and then the word $w = u_1\overline{u_2}$ represents an element $g \in G \setminus H$ such that $gHg^{-1} \cap H \neq 1$. □

Proposition 6.27. *Let $G = G_1 * \ldots * G_k$ with G_i finite or cyclic, and $\langle h_1, \ldots, h_m \rangle = H \leq_{fg} G$. we can compute H_{mal} in $O(n^3)$. Where $n = \sum_{i=1}^{m} |h_i|$.*

Proof. The proof is exactly the same as for Proposition 5.23. Using Corollary 6.26 and Lemma 5.22 we can define the sequence $H = H_0 \leq \ldots \leq H_t = H_{mal}$ with $t \leq n$. Where $H_{i+1} = \langle g_i, H_i \rangle$, with g_i the element found by Corollary 6.26. From the proof of Proposition 6.25 we know that adding this element to H_i ends up with a vertex identification in $\Gamma(H_i)$, so $|V(\Gamma(H_{i+1}))| < |V(\Gamma(H_i))|$, that is why $t \leq n$. □

6.2.1 The algorithm

In this section we will use the Product Stallings foldings explained in Section 5.2.1 to improve the time complexity of the Proposition 6.27. As we did in Section 5.2.2, we will need some extra data structures to keep them updated and to know which vertices to fold or when we are done.
Similarly to the free group algorithm, we will use the disjoint-set data structure to keep the connected components of $\Gamma(H) \times \Gamma(H)$. Also, as we are interested in colored components, for each G_i finite, we will have another disjoint-set data structure of the G_i-colored components. In this case, if a vertex is not part of any G_i-colored components we will store him as a separate component but we won't store any information about him.
Also in this case, we need to check if some components are trees or not. We will also have the table $\mathit{cycle} : V(\Gamma(H) \times \Gamma(H)) \to \mathbb{Z}$. But this time if $C \subseteq \Gamma(H) \times \Gamma(H)$ is a connected component with representative $C_r \in V(\Gamma(H) \times \Gamma(H))$, and $C_{aux} \subseteq A(\Gamma(H) \times \Gamma(H))$ is the equivalent connected component in the auxiliary graph, we will store $\mathit{cycle}(C_r) = |E^+(C_{aux})| - |V(C_{aux})| + 1$. This is because we are interested in finding cycles in the auxiliary graph.
We will have another table similar to *cycle* that will be $\mathit{graph}_i : V(\Gamma(H) \times \Gamma(H)) \to \mathbb{Z}$, there is one table for each G_i finite. For each G_i-colored component $C^i \subseteq \Gamma(H) \times \Gamma(H)$, and his representative $C^i_r \in V(\Gamma(H) \times \Gamma(H))$, we will have $\mathit{graph}_i(C^i_r) = |G_i| - |V(C^i)|$. From the previous propositions we know that when this value is 0, then $C^i \cong \mathit{Cayley}(G_i)$.
We will also have two lists, L_t and L_c, the first one will store a representative of each connected component of $\Gamma(H) \times \Gamma(H)$ that has $\mathit{cycle}(C_r) = 0$. The L_c list will store a representative of each colored component with $\mathit{graph}_i(C^i_r) \neq 0$.

To initialize the values of *cycle* we will first create the sets of the connected components in the disjoint-set data structure. Then we will compute $A(\Gamma(H) \times \Gamma(H))$ and count the number of edges and vertices in each component to get the values of *cycle*. For the graph_i we just calculate the disjoint-set data structure for each G_i and we keep in graph_i the number of vertices of each colored component. At the end with the actual value in graph_i and $|G_i|$ we get the final value. After getting the values of the table we can initialize the lists L_t and L_c.
Every time we identify the vertices (p, t) and (q, t) we have to update two different values. The first thing we have to look is if they are already in the same connected component. If they are, we will just do $\mathit{cycle}(C_r) \leftarrow \mathit{cycle}(C_r) + 1$ because we are deleting one vertex from the component. If they are from two different connected components C^p and C^q that will merge to C, we do $\mathit{cycle}(C_r) \leftarrow \mathit{cycle}(C^p_r) + \mathit{cycle}(C^q_r)$. Now every time we delete and don't copy one edge of (q, t) and this edge is labeled with a generator of an infinite factor, as we are removing one edge from the connected component of the auxiliary graph, we do $\mathit{cycle}(C_r) \leftarrow \mathit{cycle}(C_r) - 1$. One extra thing is if, for each finite G_i, (p, t) and (q, t) were part of the same G_i-colored component. If one of them was not part of any G_i-colored component the value stays the same. If both are part of the same G_i-colored component C^i we do $\mathit{graph}_i(C^i_r) \leftarrow \mathit{graph}_i(C^i_r) + 1$, and also if C is the connected component where they are, as we are removing one edge of the auxiliary graph, the one connecting (q, t) with c_{G_i}, we do $\mathit{cycle}(C_r) \leftarrow \mathit{cycle}(C_r) - 1$. The last case is when they are from different G_i-colored component $C^{i,p}$ and $C^{i,q}$, then they merge to C^i and we do $\mathit{graph}_i(C^i_r) \leftarrow \mathit{graph}_i(C^{i,p}_r) + \mathit{graph}_i(C^{i,q}_r) - |G_i| + 1$.
Every time we modify one connected component or G_i-colored component, we first take the representative out of the corresponding list, L_t or L_c, if they are in them, and then add the resulting

representatives back to the corresponding list if it is necessary.

With these data structures we can now show the final theorem:

Theorem 6.28. *Let $G = G_1 * \ldots * G_k$ where G_i are finite or cyclic groups. Let $\langle h_1, \ldots, h_m \rangle = H \leq G$. Then we can compute H_{mal} in $O(n^2 \cdot log^*(n))$, where $n = \sum_{i=1}^{m} |h_i|$.*

Proof. We will be doing the same process as in Proposition 6.27. As we did in Theorem 5.27 we will be using the data structures that we defined to go from $\Gamma(H_i) \times \Gamma(H_i)$ to $\Gamma(H_{i+1}) \times \Gamma(H_{i+1})$ and keeping them updated to decide which vertex to fold or if we finished. When checking each list L_t or L_c for elements not in the diagonal, we will have to iterate at most over $O(n)$ elements, as it is the maximum number of colored components that the diagonal component can have.
So from Proposition 5.26 and knowing that the disjoint-set data structure has an amortized time complexity of $O(log^*(n))$, we will have that the total complexity is $O(n^2 \cdot log^*(n))$. $\square$

Bibliography

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Ed. by Addison-Wesley. 1974.

[2] A. Cayley. “Desiderata and suggestions: No. 2. The Theory of groups: graphical representation”. In: *American Journal of Mathematics* 1.2 (1878), pp. 174–176.

[3] T. H. Cormen, C. E. Leierson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* Ed. by The MIT Press. Third Edition. 2009.

[4] I. Kapovich and A. Myasnikov. “Stallings foldings and subgroups of free groups”. In: *J. Algebra* 248.2 (2002), pp. 608–668.

[5] R. C. Lyndon and P. E. Schupp. *Combinatorial Group Theory.* Ed. by Springer. 1977.

[6] L. Markus-Epstein. “Stallings’ Foldings and Subgroups of Amalgams of Finite Groups”. In: *Intern. J. Algebra Comput.* 17.8 (2007), pp. 1493–1535.

[7] O. Schreier. “Die Untergruppen der freien Gruppen”. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 5.1 (1927), pp. 161–183.

[8] P. V. Silva and P. Weil. “On finite-index extensions of subgroups of free groups”. In: *Journal of Group Theory* 13.3 (2010), pp. 365–381.

[9] J. R. Stallings. “Topology of finite graphs”. In: *Invent. Math.* 71 (1983), pp. 551–565.

[10] N. W. M. Touikan. “A fast algorithm for Stallings’ folding process”. In: *Int. J. Algebra and Computation* 16.6 (2006), pp. 1031–1045.