Topological Model for Free Groups and Stalling's Folding

Fedor Prikolab

December 5 2020

Abstract

In this document I present a concise overview of the key ideas from the chapter *"Free Groups and Folding"* in *Office Hours with a Geometric Group Theorist* by Dan Margalit and Matt Clay. The focus is on two central themes: the topological model for finitely generated free groups and Stallings's folding algorithm. Using the topological viewpoint, we explore how subgroups of free groups can be understood through the geometry of graphs, gaining insight into their structure and algebraic properties. The exposition is complemented by an explicit implementation of Stallings's folding algorithm, which serves both as a computational illustration of the theory and as a practical tool for studying subgroups of free groups.

1 Introduction

Let F_n be a finitely generated free group. Given a subgroup

$$H \leq F_n$$

specified by a finite set of words, we want to understand structural properties of H. In particular, we aim to answer the following fundamental questions:

1. What is the rank of H?
2. Given a word $w \in F_n$, how can we decide whether $w \in H$?
3. Does H have finite index in F_n?
4. Is H normal in F_n?

The topological model of the fundamental group of a graph provides a powerful framework for answering these questions. In later sections, this approach leads naturally to Stallings' folding theory, which is the modern tool for algorithmic study of subgroups of free groups.

2 Topological Model for Free Groups

Throughout, a *directed graph* Γ consists of a set of vertices and a set of oriented edges. Each oriented edge e has an *initial* vertex $\iota(e)$ and a *terminal* vertex $\tau(e)$. For every edge e we include a formal inverse e^{-1} whose initial and terminal vertices are swapped.

Definition 1 (Edge Path)**.** *An* edge path *in* Γ *is a finite sequence of oriented edges*

$$\alpha = e_1 e_2 \cdots e_k$$

such that $\tau(e_i) = \iota(e_{i+1})$ *for each* i*. The empty path is allowed.*

Definition 2 (Reduced/Tight Path)**.** *An edge path* $\alpha = e_1 \cdots e_k$ *is called* tight *(or* reduced*) if it contains no backtracking, i.e.,*

$$e_{i+1} \neq e_i^{-1} \qquad \text{for all } i.$$

Every edge path can be tightened by repeatedly cancelling occurrences of ee^{-1} *and* $e^{-1}e$*.*

Definition 3 (Fundamental Group of a Graph)**.** *Let Γ be a connected directed graph and let v be a base vertex.*

A loop at v *is a tight edge path whose initial and terminal vertex is v. Denote the set of tight loops at v by $\pi_1(\Gamma, v)$.*

Concatenation of loops followed by tightening defines a binary operation

$$\alpha \cdot \beta := \mathit{tighten}(\alpha\beta),$$

which makes $\pi_1(\Gamma, v)$ into a group. This group is called the fundamental group of Γ based at v.

Definition 4 (Graph Associated to Generators)**.** *Let $F_n = \langle a_1, \ldots, a_n \rangle$. To model words and subgroups, we often consider directed graphs Γ whose edges are labelled by generator symbols a_j and their inverses a_j^{-1}.*

- *Each edge is assigned a label in $\{a_1^{\pm 1}, \ldots, a_n^{\pm 1}\}$.*
- *An edge labelled a_j is oriented in some fixed "positive" direction; an edge labelled a_j^{-1} is oriented oppositely.*

Such graphs will later be used to build subgroup graphs for $H \leq F_n$.

The Rank Formula

Theorem 1. *Let Γ be a connected finite graph with V vertices and E edges. Then for any base vertex v,*

$$\pi_1(\Gamma, v) \cong F_r, \qquad \textit{where} \qquad r = 1 - V + E.$$

Proof. Choose a spanning tree $T \subseteq \Gamma$. A tree has no cycles, so T contributes nothing to the fundamental group. Each edge of Γ not in T corresponds to an independent loop. There are $E - (V - 1)$ such edges, giving exactly

$$r = E - V + 1$$

free generators.

Formally, collapsing T to a point does not collapse any loops, and the resulting graph is a wedge of r circles, whose fundamental group is free of rank r. Thus $\pi_1(\Gamma, v) \cong F_r$. □

Remark 1. *This topological interpretation of free groups is the starting point for the theory of covering spaces of graphs and for Stallings' folding method, which gives algorithmic answers to all questions posed in the Introduction.*

3 Subgroups via Graphs

Let $H \leq F_n$ be a subgroup generated by a finite set of words. From this generating set, we construct a labelled directed graph Γ_H in which each generator word determines a loop based at a fixed vertex. There is a natural graph map

$$\Gamma_H \longrightarrow R_n,$$

where R_n is the standard *rose* with n loops labelled by the generators of F_n. This induces a homomorphism on fundamental groups

$$\rho : \pi_1(\Gamma_H, v) \longrightarrow \pi_1(R_n, v) \cong F_n.$$

Two natural questions arise:

1. How can we determine whether the induced map ρ is injective?
2. If ρ is not injective, can we algorithmically modify Γ_H so that the resulting map becomes injective?

These questions are answered by *Stallings' folding process*, which transforms Γ_H into a canonical "core graph" that faithfully represents the subgroup H inside F_n.

Folding Operations

A *fold* is a local modification of a labelled graph that identifies a pair of edges which violate the immersion condition.

Definition 5 (Fold)**.** *Let* Γ *and* Δ *be labelled directed graphs. A map* $\Phi : \Gamma \to \Delta$ *is called a* fold *if it identifies two edges*

$$e_1, e_2 \in E(\Gamma)$$

that satisfy the following:

- *they have the same initial vertex,*
- *they share the same label,*
- *and their images in* Δ *are a single edge.*

Such a fold either identifies the terminal vertices of e_1 *and* e_2*, or identifies only the edges if the terminal vertices are already the same.*

There are two qualitatively different types of folds:

- **Vertex folds:** edges with the same label are merged and their terminal vertices are identified. The induced map on π_1 is an isomorphism. (Example: two edges labelled a from a vertex u to vertices b and c are folded so that b and c merge.)
- **Edge folds:** two distinct edges with the same label and identical endpoints are merged into one. The induced map on π_1 is surjective but not injective. This eliminates redundant parallel edges and simplifies the path structure.

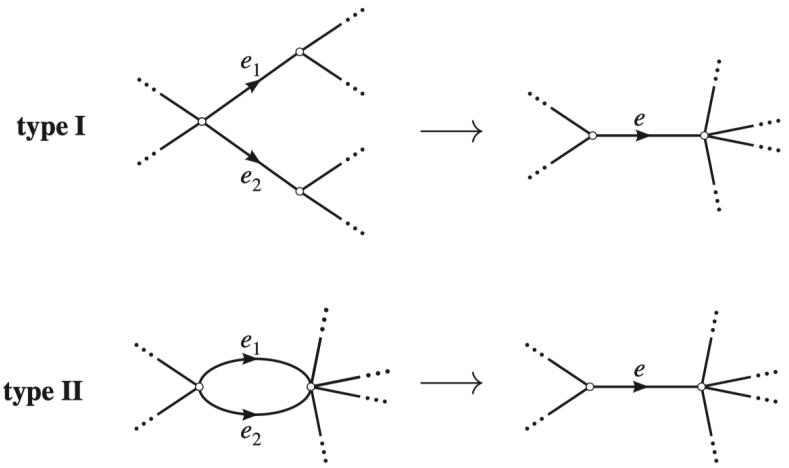

Figure 1: Two qualitatively different types of folds. Image taken from *Office Hours with a Geometric Group Theorist*.

When no further folds are possible, the resulting graph has the property that no two edges with the same label originate at the same vertex. This is the defining feature of an *immersion*.

Definition 6 (Immersion)**.** *A graph map* $\Gamma \to \Delta$ *is an* immersion *if for every vertex* $v \in \Gamma$*, the outgoing edges at* v *map injectively into the set of outgoing edges at its image in* Δ*. Equivalently, no two edges in* Γ *with the same initial vertex and the same label have the same image.*

Injectivity and Folding

Theorem 2.

1. *If $f : \Gamma \to \Delta$ is an immersion between connected labelled graphs, then the induced map on fundamental groups*
$$f_* : \pi_1(\Gamma, v) \longrightarrow \pi_1(\Delta, f(v))$$
is injective.

2. *If $f : \Gamma \to \Delta$ is any map of finite labelled graphs, then there exists a factorization*
$$\Gamma = \Gamma_0 \longrightarrow \Gamma_1 \longrightarrow \cdots \longrightarrow \Gamma_k \longrightarrow \Delta$$
where each map $\Gamma_{i-1} \to \Gamma_i$ is a fold, and the final map $\Gamma_k \to \Delta$ is an immersion.

The terminal graph Γ_k is called the *folded core* (or simply the *core graph*) of the subgroup. It is unique up to isomorphism, and the induced immersion into R_n realizes $\pi_1(\Gamma_k)$ as a subgroup of F_n. This provides an effective algorithm for computing the structure of H.

Remark 2. *The folding process removes all redundancies in the initial graph Γ_H. The resulting immersion provides a canonical labelled graph that represents the subgroup H exactly, and not merely up to surjection.*

Membership Problem

- *Question: How can we determine whether an element of F_n belongs to H?*
- *Answer:* Let Δ_H be the folded core graph obtained from Γ_H via Stallings folds. We know that
$$\pi_1(\Delta_H, v) \cong H.$$
To decide whether a word $g \in F_n$ belongs to H, begin at the base vertex v of Δ_H and attempt to read g as a tight labelled path.
  - If this path is readable and ends again at v, then $g \in H$.
  - If the path cannot be completed, or ends at a different vertex, then $g \notin H$.

  Moreover, if $g \in H$, then the path read in Δ_H expresses g as a word in the fundamental group generators of Δ_H, which correspond (after tracking folds) to words in the original generating set of H.

Subgroup Index

Question: How can we compute the index $[F_n : H]$?

A fundamental fact from covering space theory is:

The immersion $\Delta_H \to R_n$ is a covering map if and only if every vertex of Δ_H has exactly $2n$ outgoing oriented edges, one for each label $a_i^{\pm 1}$.

When this condition holds, the covering map is finite-sheeted precisely when Δ_H has finitely many vertices. In that case, every vertex of Δ_H corresponds to a distinct left coset of H in F_n, and thus:

$$[F_n : H] = |\mathrm{Vert}(\Delta_H)|.$$

Remark 3 (Example where the condition fails)**.** *Consider the inclusion of the subgraph $R_{n-1} \hookrightarrow R_n$. Interpreting R_n as the Cayley graph of F_n, the graph R_{n-1} omits one generator. Thus at each vertex, the full set of $2n$ edges is not present, so the map $R_{n-1} \to R_n$ is* not *a covering.*

In group-theoretic terms, the subgroup $\langle a_1, \ldots, a_{n-1} \rangle$ has infinite index in F_n.

Cosets from vertices Since loops based at the base vertex w of Δ_H represent exactly the subgroup H, any reduced path γ from w to a vertex w' determines an element of F_n, well-defined modulo left multiplication by H. Thus each vertex of Δ_H corresponds naturally to a left coset of H.

To see that these cosets exhaust all left cosets of H, we use the covering property:

Remark 4 (Lifting property)**.** *If $\Delta_H \to R_n$ is a covering map, then every reduced path in R_n lifts uniquely to a reduced edge path in Δ_H beginning at any chosen starting vertex.*

Justification. This is the standard path-lifting property of covering spaces. Each step of a reduced path in R_n is labelled by some generator $a_i^{\pm 1}$, and since the covering has all $2n$ edges present at each vertex, there is a unique outgoing edge with that label. Thus at each step the lift is uniquely determined, giving a unique reduced lifted path. □

The lift of any word $\delta \in F_n$ starting at w ends at some vertex $w' \in \Delta_H$, showing that $H\delta$ is one of the cosets represented by vertices of Δ_H. Since δ was arbitrary, the vertices exhaust all left cosets.

Theorem 3. *Suppose Δ_H is a finite folded graph and $\Delta_H \to R_n$ is a covering map. Then H has finite index in F_n, and*

$$[F_n : H] = |\mathrm{Vert}(\Delta_H)|.$$

Normality We now address our final structural question:

$$\textit{When is } H \trianglelefteq F_n \textit{?}$$

The subgroup H is normal in F_n if and only if

$$g^{-1}Hg \subseteq H \qquad \text{for all } g \in F_n.$$

Interpreting this in terms of the core graph Δ_H:

- A path labelled by g from w to w' corresponds to left multiplication by g.
- A loop at w' corresponds to an element of $g^{-1}Hg$.

Thus the condition $g^{-1}Hg \subseteq H$ means:

Every tight loop based at w' must also be readable as a tight loop based at w.

This is possible only if vertices w and w' support exactly the same labelled loops, i.e. the same subgroup. Since g was arbitrary, we conclude:

$$H \trianglelefteq F_n \quad \Longleftrightarrow \quad \text{all vertices of } \Delta_H \text{ support isomorphic rooted-labelled loop structures.}$$

Geometrically, this means that the group of deck transformations acts transitively on the vertices.

Definition 7 (Vertex Transitivity)**.** *A connected labelled graph Δ_H is* vertex-transitive *if for any two vertices u and v there exists a label-preserving graph automorphism of Δ_H sending u to v.*

Putting everything together:

Theorem 4. *Let $\Delta_H \to R_n$ be the folded core graph of $H \leq F_n$. Then the subgroup H is normal in F_n if and only if Δ_H is vertex-transitive.*

Remark 5. *Equivalently, H is normal if and only if the covering $\Delta_H \to R_n$ is a* regular covering*, i.e. the deck transformation group acts transitively on the vertices.*

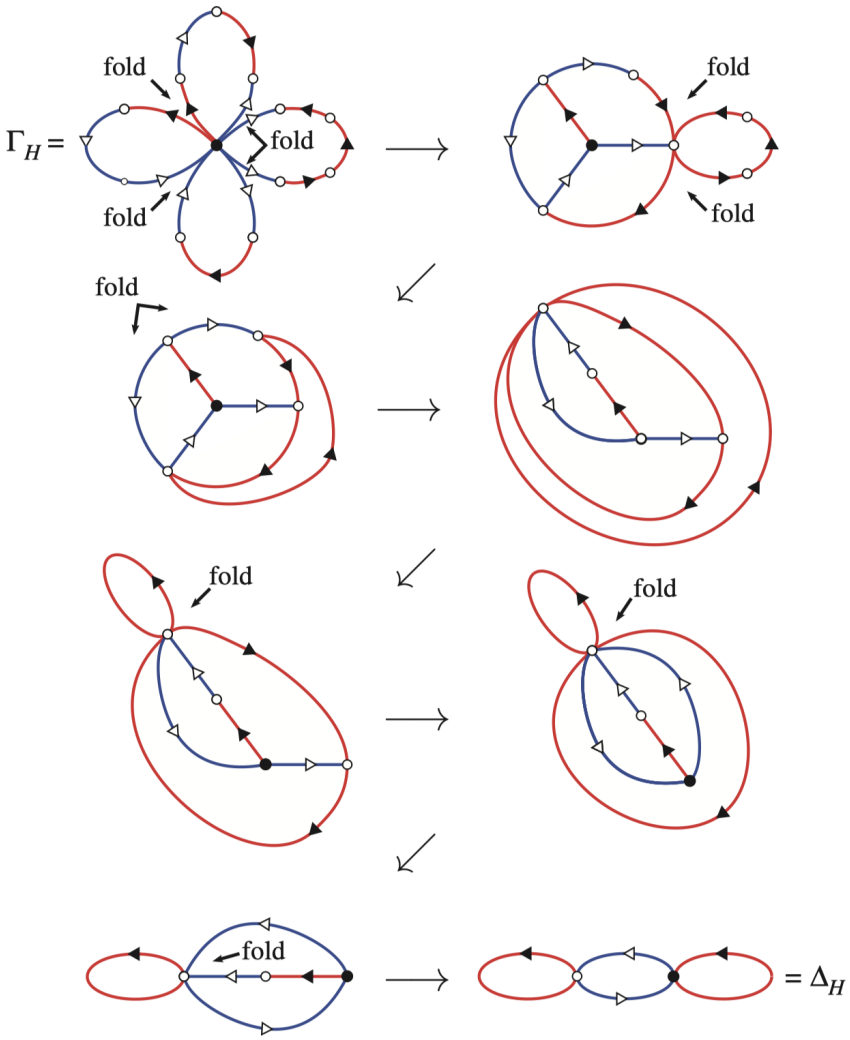

Figure 2: An example of Stallings's folding algorithm for the subgroup $H < F_2 = \langle a, b \rangle$ generated by $\langle abab^{-1}, ab^2, bab, ba^3b^{-1} \rangle$. Red edges correspond to the generator a, and blue edges correspond to the generator b. Illustration adapted from *Office Hours with a Geometric Group Theorist*.

4 Generators from a Maximal Tree

This section concludes our presentation of the Topological Model for Free Groups and Stallings's Folding with a response to a practical question: how to algorithmically find the generators of a reduced Stallings graph.

Let $F_n = \langle a_1, \ldots, a_n \rangle$ be a free group on n generators and let $H \leq F_n$ be a finitely generated subgroup. After constructing the initial graph from a set of subgroup generators and applying Stallings's folding algorithm, we obtain a finite folded graph Γ with a distinguished base vertex v_0. The immersion $\Gamma \to R_n$ into the n–petalled rose represents the inclusion $H \hookrightarrow F_n$. We now explain how to obtain a free basis for H from Γ using a maximal tree.

4.1 Maximal trees and paths from the base

We consider the underlying undirected graph of Γ. A *maximal tree* (or spanning tree) T of this undirected graph is a connected, cycle-free subgraph that contains all vertices of Γ. The tree is rooted at the base vertex v_0.

In the context of rooted trees, every vertex $v \neq v_0$ has a unique neighbor on the simple path joining v to v_0. This neighbor is called the *parent* of v, denoted parent(v). The edge joining parent(v) to v is the *tree edge of* v. The base vertex v_0 is the unique vertex with no parent.

Breadth–first search (BFS). To construct the maximal tree, our implementation performs a breadth–first search (BFS). BFS begins at the base vertex v_0 and explores all adjacent vertices before moving on to their neighbors. This process naturally produces a spanning tree: whenever BFS encounters a new vertex v for the first time, it records the current vertex as parent(v) and the connecting edge as the tree edge. Because BFS always expands layers in increasing order of distance from v_0, every vertex is discovered via a shortest path, and the resulting parent structure encodes the unique tree-path from v_0.

Given this parent structure, the unique reduced path in T from v_0 to any vertex v is found by following the chain of parents from v back to v_0, and then reversing this sequence. Reading off the labels on these edges produces a word

$$w(v_0 \to v) \in F_n,$$

which represents the path in the tree from v_0 to v.

4.2 Non-tree edges and generators of the subgroup

Every directed edge of Γ that is *not* part of the maximal tree T corresponds to a loop in Γ based at v_0. Let

$$e : u \xrightarrow{a} v$$

be a non-tree edge. We form the loop at v_0 by traversing the tree path from v_0 to u, then taking the edge e, and then returning to v_0 along the tree path from v:

$$\gamma_e = w(v_0 \to u)\, a\, w(v_0 \to v)^{-1}.$$

After free reduction, this becomes a reduced word $w_e \in F_n$. By construction, each such w_e represents an element of the subgroup H.

Independence and basis elements. Since T contains no cycles, each non-tree edge e completes *exactly one* distinct simple cycle in the graph Γ. Different non-tree edges create different cycles that cannot be written as combinations of one another using only tree edges. Algebraically, this implies that the associated reduced loops w_e represent independent elements in H. In fact, Stallings showed that the set of all such w_e, taken over all non-tree edges of Γ, forms a free basis of H. Thus the number of generators equals the number of non-tree edges, and

$$\text{rank}(H) = 1 + E - V$$

where V and E are the numbers of vertices and (undirected) edges of the core graph.

4.3 Algorithmic description

The complete method for extracting a basis from the reduced Stallings graph is as follows:

1. Begin with the fully folded graph Γ and base vertex v_0.
2. Run BFS from v_0 to produce a spanning tree T, recording each vertex's parent and tree edge.
3. For each vertex v, compute its tree path word $w(v_0 \to v)$ by following the parent chain.
4. For each directed non-tree edge $e : u \to v$ labeled by a:
   (a) compute $w(v_0 \to u)$ and $w(v_0 \to v)$;
   (b) produce the loop word
   $$w_e = \mathrm{red}\big(w(v_0 \to u)\, a\, w(v_0 \to v)^{-1}\big);$$
5. The set of all w_e is a free generating set of H.

In the accompanying Python implementation, this procedure is encoded in the methods `_spanning_tree_from_base`, `_word_base_to`, `_compute_basis_from_tree`, and made accessible through the public method `final_subgroup_generators()`, which returns the resulting basis.

References

[1] Dan Margalit and Matt Clay. *Office Hours with a Geometric Group Theorist*. Princeton University Press, 2017.