Department of Mathematics
TUM School of Computation, Information and Technology
Technische Universität München

Computation of Combinatorial Torsions

Kai Fabio Neugebauer

Wissenschaftliche Arbeit zum Erlangen des Grades Bachelor of Sciences and der TUM School of Computation, Information and Technology der Technischen Universität München

Aufgabensteller:	Prof. Dr. Ulrich Bauer
Betreuer:	Nico Stucki
Abgabetermin:	14. 07. 2023

Hiermit erkläre ich, dass ich die Bachelorarbeit selbstständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

[illegible]

München, den 14. 07. 2023

Abstract

The Reidemeister torsion is a well-known invariant in the field of algebraic topology, but is generally difficult to calculate in practice. In this thesis, we develop an algorithm that calculates the Reidemeister torsion of the geometric realization of a simplicial set. We implemented the algorithm in SageMath and used it to numerically calculate the Reidemeister torsions of the Poincaré homology 3-sphere with respect to all irreducible complex representations of its fundamental group. Similar calculations were also performed for several lens and product spaces.

Zusammenfassung auf Deutsch

Die Reidemeister-Torsion ist eine bekannte Invariante auf dem Gebiet der algebraischen Topologie. Sie lässt sich in der Praxis jedoch im Allgemeinen nur schwer berechnen. In dieser Bachelorarbeit entwickeln wir einen Algorithmus, der die Reidemeister-Torsion der geometrischen Realisierung einer simplizialen Menge berechnet. Wir haben den Algorithmus in SageMath implementiert und ihn verwendet, um die Reidemeister-Torsionen der Poincaré-Homologie-3-Sphäre in Bezug auf alle irreduziblen komplexen Darstellungen ihrer Fundamentalgruppe numerisch zu berechnen. Ähnliche Berechnungen wurden auch für mehrere Linsen- und Produkträume durchgeführt.

Contents

1. Introduction

The Reidemeister torsion was the first invariant in algebraic topology that could distinguish between CW-complexes that are homotopy equivalent but not homeomorphic. Reidemeister torsion has been extensively studied since its introduction in 1935. There is enormous interest in knowing the Reidemeister torsion of specific spaces, but in general Reidemeister torsion is very difficult to calculate by hand.

The definition of Reidemeister torsion via the chain complex of a universal cover is already kind of constructive but can be hard to handle by hand if a given CW-decomposition consists of many complicatedly arranged cells. We found a way to completely automate this process of computing Reidemeister torsion for simplicial sets with finitely many nondegenerate simplices. In particular, we were able to develop an algorithm that computes the chain complex of the universal cover of a predetermined simplicial set X as a chain-complex of free modules over the integral group ring of the fundamental group $\pi_1(X)$.

This approach to Reidemeister torsion via covering spaces lead us to consider covering projections of simplicial sets. As a byproduct of this considerations, we developed an algorithm that has input a connected simplicial set X, which has only finitely many non-degenerate simplices, and a normal, finite-index subgroup $N \subseteq \pi_1(X)$. The algorithm returns a connected simplicial set Z together with a covering projection $p : Z \to X$ such that $p_*\pi_1(X) = N$. If wanted, the covering projection $p : Z \to X$ can also be returned as the structure map of a pullback, together with the corresponding pullback square. In that case, p will be obtained as a pullback of a universal covering projection, namely the universal cover of the $\dim(X)$-skeleton of the classifying space of the quotient group $\pi_1(X)/N$.

As suggested by the title, the main goal of this bachelor thesis is to develop an algorithm to compute combinatorial torsions. By combinatorial torsion we mean, on the one hand, the torsion of chain complexes, as defined in [11, chap. 3]. But, above all, we will deal with Reidemeister torsion. Nevertheless, our approach to calculating Reidemeister torsion can not do without calculating the torsion of chain-complexes. So, we developed an algorithm that computes the torsion of a based and bounded chain-complex of finite-dimensional $\mathbb{C}$-vector spaces.

While Reidemeister torsion is defined for finite CW-complexes, we decided to work with simplicial sets as it is an implementable data structure that still offers great generality and specifically includes simplicial complexes and Δ-complexes.

We start in the first section with definitions and basic theorems on simplicial sets to settle the notations and conventions appearing in this thesis. Here, only well-known results are presented and we will for the main part follow the first chapter of [10]. Readers who are already familiar with simplicial sets are advised to skip this chapter for the time being and return only if unfamiliar notations or theorems are used later.

In the next chapter on covering space projections we introduce a convenient definition of covering projections of simplicial sets. The aim of the chapter is not to have to change categories in the remaining parts of the bachelor thesis, i.e. we want do define and compute notions like Reidemeister torsion already in the category of simplicial sets and we want to avoid having to take a detour over topological spaces or CW-complexes. Most of the results presented in this chapter can be found in [5, App. I].

The chapter on computation is divided into three parts. In the first subsection, we develop an algorithm that takes as input a simplicial set together with a subgroup of its fundamental group and returns a covering projection corresponding to that subgroup. In the second subsection, we fabricate an algorithm to compute the combinatorial torsion of a based chain complex of $\mathbb{C}$-vector spaces using singular value decomposition. Now in the third subsection, combining the results of the previous sections, we develop an algorithm that computes the Reidemeister torsion of simplicial sets.

In the last chapter we briefly discuss the data structures and relevant implementation details of the algorithms discussed in this paper.

The appendix contains the source code of the SageMath implementation of the algorithms discussed in this thesis. Some example calculations are also prepared there in a comprehensible manner. At the beginning of the appendix there is a brief overview of the implemented functions and examples. The code is also available under `https://github.com/fNeugebauer/Computation-of-Combinatorial-Torsion`.

2. Preliminaries: Simplicial Sets

2.1. The category of simplicial sets

Definition 2.1 (Simplicial sets)**.** Let Δ denote the *simplex category*, whose objects are nonempty linearly ordered sets of the form

$$[n] = \{0, 1, 2, \ldots, n\}$$

for all $n \geq 0$ and whose morphisms are (non-strictly) order-preserving functions between these sets.
A *simplicial set* X is a contravariant functor $\Delta \to \mathbf{Set}$, where $\mathbf{Set}$ is the category of sets.
We denote the presheaf category on Δ as $\mathbf{sSet}$, i.e. the objects of the category $\mathbf{sSet}$ are simplicial sets and the morphism in $\mathbf{sSet}$ are natural transformations between simplicial sets.

Terminology. Given a simplicial set X, we write X_n instead of $X([n])$. The elements of X_n are called the n-simplices of X. For a morphism, also called *map*, of simplicial sets $f : X \to Y$ we denote its component $X_n \to Y_n$ by f_n and for $\sigma \in X_n$ we sometimes write $f(\sigma)$ for $f_n(\sigma) \in Y_n$.

Remark. All small limits and colimits in $\mathbf{sSet}$ exist and are computed pointwise. A morphisms of simplicial sets $f : X \to Y$ is a mononomorphism (epimorphism) if and only if $f_n : X_n \to Y_n$ is injective (surjective) for all $n \geq 0$. f is an isomorphism if and only if $f_n : X_n \to Y_n$ is bijective for all $n \geq 0$. Pullbacks and pushforwards in the category $\mathbf{sSet}$ preserve both monomorphisms and epimorphisms.

Terminology. For n a positive integer, we denote for $i = 0, \ldots, n$

$$\delta^i := \delta^{n,i} : \ [n-1] \to [n], \quad \delta^i(j) = \begin{cases} j & j < i \\ j+1 & j \geq i \end{cases}$$

and for $n \in \mathbb{N}_0$ we denote for $i = 0, \ldots, n$

$$\sigma^i := \sigma^{n,i} : \ [n+1] \to [n], \quad \sigma^i(j) = \begin{cases} j & j \leq i \\ j-1 & j > i. \end{cases}$$

Definition 2.2 (Face and degeneracy maps)**.** For a simplicial set X we define the i^{th}-*face map* $d_{i,n} := d_i := X(\delta^{i,n}) : X_n \to X_{n+1}$ as the evaluation of the functor X on the morphism $\delta^{i,n}$, where n is any positive integer and $0 \leq i \leq n$.

We define the i^{th}-*degeneracy map* $s_{i,n} := s_i := X(\sigma^{n,i}) : X_n \to X_{n+1}$ as the evaluation of X on the morphism $\sigma^{n,i}$, where n is a non-negative integer and $0 \leq i \leq n$.

Remark. It is customary to omit the integer n in the notation of face and boundary maps if the domain is clear.

For any simplicial set X the degeneracy maps and face maps satisfy the *simplicial identities*:

1. For $n \geq 2$ and $0 \leq i < j \leq n$
$$d_i \circ d_j = d_{j-1} \circ d_i \tag{1}$$
as maps X_n to X_{n-2}.

2. For $n \geq 0$ and $0 \leq i \leq j \leq n$
$$s_i \circ s_j = s_{j+1} \circ s_i \tag{2}$$
as maps $X_n \to X_{n+2}$.

3. For $n \geq 0$ and $0 \leq i, j \leq n$
$$d_i \circ s_j = \begin{cases} s_{j-1} \circ d_i & \text{if } i < j \\ \mathrm{id}_{C_n} & \text{if } i = j \text{ or } i = j+1 \\ s_j \circ d_{i-1} & \text{if } i > j+1. \end{cases} \tag{3}$$
as maps $X_n \to X_n$.

Conversely, given a sequence $\{X_n\}_{n \geq 0}$ of sets together with maps $d_{i,n} : X_n \to X_{n-1}$ for $n \geq 1$, $0 \leq i \leq n$ and $s_{i,j} : X_n \to X_{n+1}$ for $n \geq 0$, $0 \leq i \leq n$ satisfying the simplicial identities there is a unique simplicial set X that has these face maps and degeneracy maps.

Remark 2.3. Let X be a simplicial set and for every $n \geq 0$ let $Y_n \subseteq X_n$ be a subset, such that the face maps $d_i : X_n \to X_{n-1}$ and degeneracy maps $\sigma_i : X_n \to X_{n+1}$ of X carry Y_n to Y_{n-1} and Y_{n+1}, respectively. Then the collection $\{Y_n\}_{n \geq 1}$ inherits the structure of a simplicial set Y. We will say that Y is a *simplicial subset* of X and the morphism $\iota : Y \to X$ with $\iota_n(\sigma) = \sigma$ for all $n \geq 0, \sigma \in X_n$ is called the *inclusion* of Y into X. Note that ι is a monomorphism. Further, (X, Y) is called a *pair of simplicial sets.*

Construction 2.4 (Preimage and image)**.** Given a morphism of simplicial sets $f : X \to Y$ and simplicial subsets $X' \subseteq X$, $Y' \subseteq Y$. We define $f|_{X'}$ as the composition $X' \hookrightarrow X \xrightarrow{f} Y$. The *image* $f(X')$ of X' under f is defined as the simplicial subset of Y with $f(X')_n = f(X'_n)$ for all $n \geq 0$. Then $f(X')$ together with the monomorphism given by the inclusion morphism $f(X') \hookrightarrow Y$ satisfies the following universal property:

- There exists a morphism, by abuse of notation, also denoted by $f : X' \to f(X')$, such that post-composing it with the inclusion morphism $f(X') \hookrightarrow Y$ equals the original f.
- For any simplicial set Z with a morphism $e : X \to Z$ and a monomorphism $m : Z \to Y$ such that $m \circ e = f$ there exists a unique morphism $v : f(X') \to Z$ such that the composition $m \circ v$ equals the inclusion of $f(X')$ into X.

Further, we define the preimage of Y' under f, denoted $f^{-1}(Y')$, as the simplicial subset of X given by $f^{-1}(Y')_n = f_n^{-1}(Y'_n)$ for all $n \geq 0$. We, by abuse of notation, also denote the map $f^{-1}(Y') \to Y'$, which is induced by f, as $f|_{f^{-1}(Y')}$. Then $f^{-1}(Y')$ together with the inclusion of $f^{-1}(Y')$ into X and $f|_{f^{-1}(Y')}$ satisfy the universal property of the pullback of the diagram $Y' \hookrightarrow Y \xleftarrow{f} X$.

Definition 2.5 (The standard simplex and its simplicial subsets)**.**

a) For $n \geq 0$ we define the standard n-simplex Δ^n to be the contravariant Hom-functor $\mathrm{Hom}_\Delta(-, [n])$, considered as a simplicial set. By convention, we set $\Delta^{-1} = \emptyset$. We define the *geometric realization* of Δ^n to be the topological space $|\Delta^n|$ given by the convex hull of the standard basis of $\mathbb{R}^{n+1}$ equipped with the subspace topology from $\mathbb{R}^{n+1}$. To each $\alpha \in \mathrm{Hom}_\Delta([m], [n])$ we assign $|\alpha| \in \mathrm{Hom}_{\mathbf{Top}}(|\Delta^m|, |\Delta^n|)$, where

$$|\alpha|(t_0, \dots, t_m) \mapsto \left(\sum_{\alpha(i)=0} t_i, \sum_{\alpha(i)=1} t_i, \dots, \sum_{\alpha(i)=n} t_i \right).$$

b) For $n \geq 0$ an integer and $\mathcal{U}$ a nonempty collection of subsets of $[n] = \{1, \dots, n\}$ we will say that $\mathcal{U}$ is downward closed if $\emptyset \neq I \subseteq J \in \mathcal{U}$ implies that $I \in \mathcal{U}$. If this condition is satisfied, we let $\Delta^n_{\mathcal{U}}$ be the simplicial subset of Δ^n whose m-simplices are non-decreasing maps $\alpha : [m] \to [n]$ such that $\mathrm{im}(\alpha) \in \mathcal{U}$. We set

$$|\Delta^n_{\mathcal{U}}| = \{(t_0, \dots, t_n) \in |\Delta^n| : \{i \in [n] : t_i \neq 0\} \in \mathcal{U}\}.$$

c) For each $n \geq 0$ the boundary $\partial\Delta^n$ is defined as $\Delta_{\mathcal{U}}$, where $\mathcal{U}$ is the collection of all nonempty proper subsets of $[n]$.

d) For $0 \leq i \leq n$ the horn Λ^n_i is defined as $\Delta^n_{\mathcal{U}}$, where $\mathcal{U}$ is the collection of all subsets of $[n]$ distinct from $[n]$ and $[n] \setminus \{i\}$.

Remark 2.6 (Yoneda lemma)*.* For each $n \geq 0$, the pair $(\Delta^n, \mathrm{id}_{[n]} \in \Delta^n_n)$ satisfies the following universal property: For any simplicial set X and n-simplex $\sigma' \in X_n$ there exists a unique morphism $\sigma \in \mathrm{Hom}_{\mathbf{sSet}}(\Delta^n, X)$ with $\sigma_n(\mathrm{id}_{[n]}) = \sigma'$. Concretely, σ is given by $\sigma_m(\alpha) = X(\alpha)(\sigma')$ for $\alpha \in \Delta^n_m$. It is customary to identify σ and σ'. In particular, we denote morphisms $\Delta^n \to \Delta^m$ as α for $\alpha \in \mathrm{Hom}_\Delta([n], [m])$ the unique order-preserving map such that the morphism $\Delta^n \to \Delta^m$ is given by post-composition with α. For σ an n-simplex of a simplicial set X and for every morphism $\alpha : [n] \to [m]$ in Δ, the morphism corresponding to $X(\alpha)(\sigma)$ factors as $\Delta^m \xrightarrow{\alpha} \Delta^n \xrightarrow{\sigma} X$. In particular, $d_i(\sigma) = \sigma \circ \delta^{i,n}$ and $s_i(\sigma) = \sigma \circ \sigma^{n,i}$, whenever those notions are defined.

Proposition 2.7 (Characterizations of horns)**.** *[10, Exercise 000Z] Let X be a simplicial set, m a non-negative integer and $0 \leq i \leq m$. Then*

$$\mathit{Hom}_{\mathbf{sSet}}(\Lambda^m_i, X) \to \prod_{j=0}^{m-1} X_{m-1}, \quad f \mapsto \{f \circ \delta^j\}_{j \in [m] \setminus \{i\}}$$

is injective with image the collection of "incomplete" sequences $(\sigma_1, \dots, \sigma_{i-1}, \cdot, \sigma_i, \dots, \sigma_n)$ *with* $d_k(\sigma_j) = d_{j-1}(\sigma_k)$ *for* $j, k \in [n] \setminus \{i\}$ *and* $k < j$*. (In particular, for* $j \in [m] \setminus \{i\}$*, the morphism* $\delta^j : \Delta^{m-1} \to \Delta^m$*, factors as composition* $\Delta^{m-1} \to \Lambda^m_i \hookrightarrow \Delta^m$*, where we, by abuse of notation, denote the first map as* δ^j*, also.)*

Definition 2.8 (A vertex of a simplex)**.** Let X be a simplicial set X, n a non-negative integer, $i \in [n]$ and $\mathcal{U} \subseteq [n]$ a downward closed collection of subsets of $[n]$, such that $\{i\} \in \mathcal{U}$. For any $\sigma : \Delta^n_{\mathcal{U}} \to X$, we define *the* i^{th} *vertex of* σ, denoted $\sigma(i)$, as the composition $\Delta^0 \xrightarrow{\alpha} \Delta^n_{\mathcal{U}} \xrightarrow{\rho} X$ for $\alpha : [0] \to [n]$, $0 \mapsto i$. Clearly, $\alpha : \Delta^0 \to \Delta^n$ has image in $\Delta^n_{\mathcal{U}}$ and therefore $\sigma(i) \in X_0$ is a well-defined 0-simplex of X.

2.2. Nondegenerate simplices

Proposition 2.9. *[10, Tag 0010] Let* X *be a simplicial set and let* $\sigma \in X_n$ *for some* $n > 0$*, which we identify with a map of simplicial sets* $\sigma : \Delta^n \to X$*. The following conditions are equivalent:*

a*) The simplex* σ *belongs to the image of the degeneracy map* $s_i : X_{n-1} \to X_n$ *for some* $0 \leq i \leq n-1$*.*

b*) The map* σ *factors as a composition* $\Delta^n \xrightarrow{\alpha} \Delta^{n-1} \to \Delta^n$*, where* α *is a surjective map of linearly ordered sets* $[n] \to [n-1]$*.*

c*) The map* σ *factors as a composition* $\Delta^n \xrightarrow{\alpha} \Delta^m \to X$*, where* $m < n$ *and* α *is a surjective map of linearly ordered sets* $[n] \to [m]$*.*

d*) The map* σ *factors as a composition* $\Delta^n \to \Delta^m \to X$*, where* $m < n$*.*

e*) The map* σ *factors as* $\Delta^n \xrightarrow{\sigma'} \Delta^m \to X$*, where* σ'_0 *is not injective.*

Definition 2.10 (Degenerate and nondegenerate simplices)**.** Let X be a simplicial set and let $\sigma : \Delta^n \to X$ be an n-simplex. σ is called *degenerate* if $n > 0$ and σ satisfies the equivalent conditions from proposition 2.9. We say that σ is *nondegenerate* if σ is not degenerate. For each $n \geq 0$ we let $X^{\text{nd}}_n \subseteq X_n$ denote the subset of X_n consisting of nondegenerate n-simplices of X.

In the next proposition, we cite two results that explain why in an implementation of simplicial sets only non-degenerate simplices have to be specified.

Proposition 2.11 (Simplicial sets are determined by nondegenerate simplices)**.**

a*) Let* $f : X \to Y$ *be a morphism of simplicial sets. If* σ *is a degenerate* n*-simplex of* X*, then* $f(\sigma)$ *is a degenerate* n*-simplex of* Y*.*

b*) Let* $\sigma : \Delta^n \to X$ *be an* n*-simplex of* X*. Then* σ *can be factored as composition*

$$\Delta^n \xrightarrow{\alpha} \Delta^m \xrightarrow{\tau} X,$$

where α is a surjective map of linearly ordered sets $[n] \to [m]$ and τ is a nondegenerate m-simplex of X. Moreover, this factorisation is unique and m is the smallest nonnegative integer for which σ can be factored as a composition $\Delta^n \to \Delta^m \to X$.

c) *A morphism $f : X \to Y$ of simplicial sets is a monomorphisms if and only if f maps any two distinct nondegenerate simplices of X to distinct nondegenerate simplices of Y.*

Proof. a) is clear and b) is [10, Proposition 0014]. For c), the only if part follows by 2.1 and the definition of degeneracy. For the other direction, it is straightforward to prove that f_n is injective for all n by hand using part b) of this proposition. For a different proof see [19, Lemma 017S]. □

Convention 2.12. Let X be a simplicial set and let $\sigma : \Delta^n \to X$ be an n-simplex. By proposition 2.11 and the simplicial identity (2), there exists a unique triple (m, τ, I), where $m \geq 0$ is an integer, τ is a nondegenerate m-simplex of X and $I = \{i_1, \dots, i_{n-m}\}$ with $n > i_1 > i_2 > \cdots > i_{n-m} \geq 0$, such that

$$s_I(\tau) := s_{i_1} \circ s_{i_2} \circ \cdots \circ s_{i_{n-m}}(\tau) = \sigma.$$

From now on, the notation s_I will be reserved for exactly this use. Further, we define

$$\sigma^I = \sigma^{i_{n-m}} \circ \sigma^{i_{n-m-1}} \circ \cdots \circ \sigma^{i_2} \circ \sigma^{i_1} : [n] \to [m],$$

which is the morphism in Δ such that $s_I = X(\sigma^I)$ for any simplicial set X. Be careful, σ^I is an n-simplex of Δ^m but not a simplex of X. In fact, $\sigma : \Delta^n \to X$ equals the composition $\Delta^n \xrightarrow{\sigma^I} \Delta^m \xrightarrow{\tau} X$.

2.3. The skeletal filtration

Definition 2.13 (Dimension of simplicial sets)**.** For $k \geq -1$ a simplicial set X has *dimension* $\leq k$, if for all $n > k$ every n-simplex in X_n is degenerate. If $k \geq 0$, we say X has *dimension* k if X has dimension $\leq k$ but not dimension $\leq k - 1$. We say X is *finite dimensional* if there exists $k \geq 0$ such that $\dim X \leq k$. Further, we say that X is of *finite type*, if for all $n \geq 0$ the set X_n has finite cardinality. X is called *finite*, if X is finite dimensional and of finite type.

Construction 2.14 (The k-skeleton)**.** Let X be a simplicial set and let $k \geq -1$ be an integer. It follows from proposition 2.11 b) that for any n-simplex $\sigma : \Delta^n \to X$ of X the following are equivalent:

a) $\sigma = s_I(\tau)$ for $\tau \in X_m^{\mathrm{nd}}$. Then $m \leq k$.

b) There exists a factorization $\Delta^n \to \Delta^{m'} \to X$ of σ with $m' \leq k$.

For each $n \geq 0$, we denote by $\mathrm{sk}_k(X_n)$ the subset of X_n containing all simplices of X_n satisfying conditions a) and b) above. From b) and remark 2.6 we see that the collection $\{\mathrm{sk}_k(X_n) \subseteq X\}_{n\geq 0}$ is stable under the face and degeneracy operators of X and therefore determines a simplicial subset of X by remark 2.3. We will call this simplicial subset the *k-skeleton* of X and denote it by $\mathrm{sk}_k(X)$.
If $n \leq k$ then $\mathrm{sk}_k(X)_n = \mathrm{sk}_k(X_n) = X_n$, in particular,

$$X = \bigcup_{k\geq -1} \mathrm{sk}_k(X). \tag{4}$$

If $n > k$, then $\mathrm{sk}_k(X)_n = \mathrm{sk}_k(X_n) \subseteq X_n \setminus X_n^{\mathrm{nd}}$. So, $\dim(\mathrm{sk}_k(X)) \leq k$ for all $k \geq 0$. Also, $\mathrm{sk}_{-1}(X) = \emptyset$.

Remark. Some authors define sk_k as a functor from **sSet** to $\mathbf{sSet}_{\leq k}$, which is the category of functors from $\Delta_{\leq k}$ to **Set**, where $\Delta_{\leq k}$ is the full subcategory of Δ with objects $[m]$ for $m \leq k$.

Proposition 2.15 (Universal property of the k-skeleton). *[10, Tag 001A] The k-skeleton of a simplicial set X and its inclusion $sk_k(X) \hookrightarrow X$ satisfies the following universal property: For every simplicial set Y of dimension $\leq k$, post-composition with the inclusion $sk_k(X) \hookrightarrow X$, induces a bijection*

$$Hom_{\mathbf{sSet}}(Y, sk_k(X)) \to Hom_{\mathbf{sSet}}(Y, X).$$

Remark. In fact, the functor $\mathrm{sk}_k : \mathbf{sSet} \to \mathbf{sSet}$ admits a left adjoint, the so-called *co-skeleton* functor, see [19, Tag 018K].

Proposition 2.16 (Nondegenerate simplices of the product). *For two simplicial sets X, Y and $n \geq 0$ a non-negative integer let $\Psi : (X \times Y)_n \to X_n \times Y_n$ be the canonical bijection induced by the projections on the factors. Then Ψ precomposed with the inclusion $(X \times Y)_n^{nd} \hookrightarrow (X \times Y)_n$ has image*

$$\{(s_I(\sigma), s_J(\tau)) \in X_n \times Y_n \mid \sigma, \tau \text{ nondegenerate and } I \cap J = \emptyset\}.$$

If σ and τ are nondegenerate simplices of dimensions k and l, in the product they will lead to $\binom{n}{k} \cdot \binom{k}{n-l}$ nondegenerate simplices in dimension $n \leq k + l$ and no nondegenerate simplices in dimension $n > k+l$. In particular, $\dim(X \times Y) = \dim X + \dim Y$, whenever X and Y are nonempty.

Proof. This is follows from the simplicial identity (2) and the fact $s_i(\Psi^{-1}(s_I(\sigma), s_J(\sigma))) = \Psi^{-1}(s_i(s_I(\sigma)), s_i(s_J(\tau)))$ for $i = 0, \ldots, r$ and any $r \geq 0$. See [4, Example 5.5]. □

Definition 2.17 (Attaching map). Let X be a simplicial set and $k \geq 0$. Let $\sigma \in X_k^{\mathrm{nd}}$. Since the boundary $\partial\Delta^k \subseteq \Delta^k$ has dimension $k-1$, the composition $\partial\Delta^k \hookrightarrow \Delta^k \xrightarrow{\sigma} X$ equals the composition $\partial\Delta^k \xrightarrow{\phi_\sigma} \mathrm{sk}_{k-1}(X) \hookrightarrow X$ for a unique $\phi_\sigma : \partial\Delta^k \to \mathrm{sk}_{k-1}(X)$. This is by proposition 2.15. We refer to ϕ_σ as the attaching map of σ. Further, we call the unique map $\Phi_\sigma : \Delta^k \to \mathrm{sk}_k(X)$ such that σ factors as composition $\Delta^k \xrightarrow{\Phi_\sigma} \mathrm{sk}_k(X) \hookrightarrow X$ the characteristic map of σ.

Proposition 2.18 (The skeletal filtration). *Let X be a simplicial set and let $k \geq 0$. Then the following diagram is a pushout square in the category* ***sSet****:*

$$\begin{tikzcd}
\coprod_{\sigma \in X_k^{nd}} \partial\Delta^k \arrow[r] \arrow[d, "(\phi_\sigma)"] & \coprod_{\sigma \in X_k^{nd}} \Delta^k \arrow[d, "(\Phi_\sigma)"] \\
sk_{k-1}(X) \arrow[r] & sk_k(X)
\end{tikzcd}$$

The top horizontal map is the coproduct of inclusions $\partial\Delta^k \hookrightarrow \Delta^k$ indexed over X_k^{nd} and the lower horizontal map is the canonical inclusion $sk_{k-1}(X) \hookrightarrow sk_k(X)$.

Proof. The commutativity of the diagram, which is immediate from definition 2.17, is equivalent to the existence of a map F from the pushout to $\mathrm{sk}_k(X)$. Let $\sigma : \Delta^n \to \mathrm{sk}_k(X)$ be an n-simplex of $\mathrm{sk}_k(X)$. It suffices to prove that $\sigma \in s_k(X_n)$ has a unique preimage under F. By proposition 2.11 b) there is a unique pair of a surjective map of linearly ordered sets $\alpha : [n] \to [m]$ and a nondegenerate m-simplex $\tau : \Delta^m \to \mathrm{sk}_k(X)$, such that σ equals to the composition $\Delta^n \xrightarrow{\alpha} \Delta^m \xrightarrow{\tau} \mathrm{sk}_k(X)$. By definition of the k-skeleton 2.14 we must have $m \leq k$. σ has a preimage in $\mathrm{sk}_{k-1}(X)$ if and only if $m < k$. So, if $m = k$ the statement is clear. When $m < k$, then τ factors uniquely through a non-degenerate m-simplex λ of $\mathrm{sk}_{k-1}(X)$. Also, every nondegenerate m-simplex $\rho : \Delta^m \to \Delta^k$ with $\Phi_\sigma \circ \rho = \tau$ factors through $\partial\Delta^k$ and is therefore identified with λ in the pushout. □

2.4. Geometric realization

Construction 2.19 (The singular simplicial set of a topological space). For a topological space S we define a simplicial set $\mathrm{Sing}(S)$, the so-called *singular simplicial set* of S, as follows: Each $[n]$ will be assigned to $\mathrm{Hom}_{\mathbf{Top}}(|\Delta^n|, S)$ and each $\alpha \in \mathrm{Hom}_\Delta([n],[m])$ will be assigned to precomposition by $|\alpha|$, where $|\alpha|$ was defined in definition 2.5. $\mathrm{Sing} : \mathbf{Top} \to \mathbf{sSet}$ becomes a functor by assigning to a continuous map $f : S \to T$ the natural transformation $f_\#$ that is given by post-composition by f on every component.

Definition 2.20 (Geometric Realization). Let X be a simplicial set and S a topological space. A map of simplicial sets $u_X : X \to \mathrm{Sing}(S)$ is said to *exhibit S as a geometric realization of X* if, for every topological space T, the composite map

$$\mathrm{Hom}_{\mathbf{Top}}(S,T) \xrightarrow{\mathrm{Sing}(-)} \mathrm{Hom}_{\mathbf{sSet}}(\mathrm{Sing}(S), \mathrm{Sing}(T)) \xrightarrow{u^*} \mathrm{Hom}_{\mathbf{sSet}}(X, \mathrm{Sing}(T))$$

is bijective, where u_X^* denotes precomposition by u_X.

If $u_X : X \to \mathrm{Sing}(S)$, $u_Y : Y \to \mathrm{Sing}(T)$ exhibit S and T as geometric realization of X and Y, respectively, and $f \in \mathrm{Hom}_{\mathbf{sSet}}(X,Y)$ then we define $|f| \in \mathrm{Hom}_{\mathbf{Top}}(S,T)$ as the unique continuous map with $|f|_\# \circ u_X = u_Y \circ f$.

By this assignments we are planning to obtain a *geometric realization functor* $|\cdot| : \mathbf{sSet} \to \mathbf{Top}$, which sends a simplicial set X to a geometric realization $|X|$ and a morphism f to $|f|$. Then, $|\cdot|$ is defined up to a natural isomorphism. By [12, Prop. 2.10], it suffices to show that every simplicial set X admits a geometric realization, and then $|\cdot|$ will automatically promote to a functor that is left-adjoint to $\mathrm{Sing}(-)$.

Lemma 2.21. *[10, Proposition 0028] For n a non-negative integer and $\mathcal{U}$ a downward closed collection of subsets of $[n]$, the canonical map $\Delta^n \to Sing(|\Delta^n|)$ determined by $id_{|\Delta^n|} \in Sing_n(|\Delta^n|)$ restricts to a map of simplicial sets $\Delta^n_{\mathcal{U}} \to Sing(|\Delta^n_{\mathcal{U}}|)$, which exhibits $|\Delta^n_{\mathcal{U}}|$ as geometric realization of $\Delta^n_{\mathcal{U}}$. $|\Delta^n_{\mathcal{U}}$ was defined in definition 2.5. For $\alpha \in Hom_\Delta([m],[n])$, the map $|\alpha|$ from definition 2.5 satisfies $|\alpha|_\# \circ u_{\Delta^m} = u_{\Delta^n} \circ \alpha$, which asserts that our notation is consistent, in the sense that $\alpha : \Delta^m \to \Delta^n$ has geometric realization $|\alpha|$.*

Proof. For $\Delta^n_{\mathcal{U}} = \Delta^n$ this follows directly from definition of $\mathrm{Sing}(-)$ and the Yoneda lemma. The lemma can then be proven by induction on the cardinality of $\mathcal{U}$, see [10, Proposition 0028]. The last statement is a straightforward check. □

Lemma 2.22. *[10, Lemma 0023] Let J be a small category and $F : J \to$ **sSet** a functor. Let $X = colim_{j\in J}F(j)$ be a colimit of F. If each of the simplicial sets $F(j)$ admits a geometric realization $|F(j)|$, then X admits a geometric realization, given by the colimit $|X| = colim_{j\in J}|F(j)|$.*

Proof. In view of definition 2.20, this is a reformulation of "left adjoints preserve colimits", for details see [10, Tag 0023]. □

Construction 2.23 (Existence of geometric realization)**.** Given a simplicial set X. We use induction on $k \geq -1$ to prove that $\mathrm{sk}_k(X)$ admits a geometric realization.
The empty map exhibits the empty set as geometric realization of $\mathrm{sk}_{-1}(X)$. Suppose for given $k \geq 0$, we have already constructed a topological space $|\mathrm{sk}_{k-1}(X)|$ and a map $u_{\mathrm{sk}_{k-1}(X)} : \mathrm{sk}_{k-1}(X) \to \mathrm{Sing}(|\mathrm{sk}_{k-1}(X)|)$ that exhibits $|\mathrm{sk}_{k-1}(X)|$ as geometric realization of $\mathrm{sk}_{k-1}(X)$. Then, we define $|\mathrm{sk}_k(X)|$ via the following pushout in the category **Top**

$$\begin{tikzcd}
\coprod_{\sigma\in X_k^{\mathrm{nd}}} |\partial\Delta^k| \arrow[r] \arrow[d, "(|\phi_\sigma|)"] & \coprod_{\sigma\in X_k^{\mathrm{nd}}} |\Delta^k| \arrow[d, "\Phi"] \\
|\mathrm{sk}_{k-1}(X)| \arrow[r] & |\mathrm{sk}_k(X)|
\end{tikzcd} \tag{5}$$

and define a morphism of simplicial sets $u_{\mathrm{sk}_k(X)} : \mathrm{sk}_k(X) \to \mathrm{Sing}(|\mathrm{sk}_k(X)|)$ using the universal property of the pushout in 2.18 as follows: On $\mathrm{sk}_{k-1}(X)$ the map should equal $u_{\mathrm{sk}_{k-1}}$. Further, each non-degenerate k-simplex τ of $\coprod_{\sigma\in X_k^{\mathrm{nd}}} \Delta^k$ should be send to the restriction of Φ to the component of $\coprod_{\sigma\in X_k^{\mathrm{nd}}} |\Delta^k|$ corresponding to τ. That $u_{\mathrm{sk}_k(X)}$ is well-defined and exhibits $|\mathrm{sk}_k(X)|$ as a geometric realization of $\mathrm{sk}_k(X)$ follows from lemma 2.22 in conjunction with proposition 2.18. The same lemma proves that $\Phi = (|\Phi_\sigma|)_{\sigma\in X_k^{\mathrm{nd}}}$ for Φ_σ the characteristic map of $\sigma \in X_k^{\mathrm{nd}}$.
Define $|X| := \mathrm{colim}_{k\geq 0}|\mathrm{sk}_k(X)|$ as the colimit in the category of topological spaces. By equation (4) and lemma 2.22 the unique map $u_X : X \to \mathrm{Sing}|X|$, that agrees on $\mathrm{sk}_k(X)$ with $u_{\mathrm{sk}_k(X)}$ for all $k \geq 0$, exhibits $|X|$ as a geometric realization of X.

Remark/Definition 2.24. Let X be a simplicial set. $|X|$ admits the structure of a CW-complex $|X|^{\mathbf{CW}}$ with skeletal filtration

$$|\mathrm{sk}_{-1}(X)| \subseteq |\mathrm{sk}_0(X)| \subseteq |\mathrm{sk}_1(X)| \subseteq \cdots \subseteq X.$$

Given a morphism $f : X \to Y$ of simplicial sets and $\sigma \in X_n$ for $n \geq 0$. Then $\sigma = s_J(\rho)$ for a unique $\rho \in X_k^{\mathrm{nd}}$ with $k \leq n$ and $f(\rho) = s_I(\tau)$ for a unique $\tau \in Y_m^{\mathrm{nd}}$ with $m \leq k$.

a) The k-simplex $u_X(\rho) \in \mathrm{Sing}(X)_k$ is the geometric realization of $\Delta^k \xrightarrow{\rho} X$, written $|\rho|$, and equals the composition $|\Delta^k| \xrightarrow{|\Phi_\rho|} |\mathrm{sk}_k(X)| \hookrightarrow |X|$. We will call

$$e_\rho := |\rho|(|\Delta^n| \setminus |\partial\Delta^n|)$$

the open cell of $|X|^{\mathbf{CW}}$ corresponding to ρ.

b) The n-simplex $u_X(\sigma) \in \mathrm{Sing}(X)_n$ is the geometric realization of $\Delta^n \xrightarrow{\sigma} X$, written $|\sigma|$, and equals $s_J(u_x(\sigma))$, which, in turn, equals the composition $|\Delta^n| \xrightarrow{|\sigma^J|} |\Delta^k| \xrightarrow{|\Phi_\rho|} |\mathrm{sk}_k(X)| \hookrightarrow |X|$. By abuse of notation, we denote the image of the map $|\sigma| : |\Delta^n| \to |X|$ by $|\sigma| \subseteq |X|$.

c) We have that $u_Y(f(\rho)) = |f| \circ |\rho|$, which is given by the composition

$$|\Delta^k| \xrightarrow{|\Phi_\rho|} |\mathrm{sk}_k(X)| \hookrightarrow |X| \xrightarrow{|f|} Y.$$

Further, $|f| \circ |\rho|$ equals $u_Y(s_I(\tau))) = |\tau| \circ |\sigma^I|$, which is given by the composition

$$|\Delta^k| \xrightarrow{|\sigma_I|} |\Delta^m| \xrightarrow{|\Phi_\tau|} |\mathrm{sk}_m(Y)| \hookrightarrow |Y|,$$

using the notation σ^I from the convention 2.12.

d) $|f|\circ|\sigma| = |f|\circ|\rho\circ\sigma^J| = |f|\circ|\rho|\circ|\sigma^J| = |\tau|\circ|\sigma^I|\circ|\sigma^J| = |\tau\circ\sigma^I\circ\sigma^J| = |(s_J\circ s_I)(\tau)|$.

In particular, $|f|$ is filtration preserving, i.e. a cellular map, and therefore the functor $|\cdot|$ factors as $\mathbf{sSet} \xrightarrow{|\cdot|^{\mathbf{CW}}} \mathbf{CW} \hookrightarrow \mathbf{Top}$.

Remark 2.25 (Geometric realization and limits). The geometric realization functor $\mathbf{sSet} \to \mathbf{Top}$, in general, preserves equalizers, see [13]. Given simplicial set X and Y, then $|X \times Y|$ is canonically homeomorphic to $Z := (|X| \times |Y|)_C$, where Z has underlying set $|X| \times |Y|$ and $A \subseteq Z$ is closed if and only if $A \cap K$ is closed in K for every compact subspace K of $|X| \times |Y|$. When either X or Y is locally finite or has only countably many nondegenerate simplices, then the canonical continuous map $Z \to |X| \times |Y|$ is a homeomorphism. This is due to [8, Prop. A.15]. Here a simplicial set X is called locally finite, when for all 0-simplices $v \in X_0$ there exist only finitely many non-degenerate simplices of X having v as a vertex. A locally finite simplicial set has a locally finite realization as a CW-complex. Locally finite CW-complexes are precisely those with locally compact underlying topological space. So, for any simplicial set X we have that $|X|$ is locally compact if and only if X is locally finite.

Proposition 2.26. *The geometric realization* $\boldsymbol{sSet} \to \boldsymbol{Top}$ *is a faithful, conservative functor.*

Proof. Given a simplicial set X. We prove that $u_X : X \to \mathrm{Sing}(|X|)$ is a monomorphism by appealing to proposition 2.11 c). Let $\sigma, \tau \in X_n^{\mathrm{nd}}$ for $n \geq 0$ such that $u_X(\sigma) = u_Y(\rho)$. By remark 2.24 a), we immediately get that $|\Phi_\rho| = |\Phi_\tau|$. But $|\Phi_\rho|$ and $|\Phi_\tau|$ are components of the map Φ from diagram (5) and therefore $\sigma = \tau$. Now suppose for the sake of contradiction, there exists $g : |\Delta^{n-1}| \to |X|$ and $0 \leq i \leq n-1$ with $s_i(g) = u_X(\sigma) \in \mathrm{Sing}_n(|X|)$. Recall the terminology 2.1, where we defined $\sigma^i : [n] \to [n-1]$. Then $g \circ |\sigma^i| = |\sigma| : |\Delta^n| \to |X|$ by definition of the functor $\mathrm{Sing}(-)$. In particular, since $|\sigma^i|$ is surjective, g factors as composition $|\Delta^{n-1}| \xrightarrow{g'} |\mathrm{sk}_n(X)| \hookrightarrow |X|$ for some continuous g'. Then $g' \circ |\sigma^i| = |\Phi_\sigma| : |\Delta^n| \to |\mathrm{sk}_n(X)|$ by remark 2.24. But in view of diagram (5) and the comments below, $|\Phi_\sigma|$ is injective on the interior of $|\Delta^n|$, whereas $|\sigma^i|$ collapses $|\Delta^n|$ to its i-th face. This contradiction implies that $u_X(\sigma)$ is nondegenerate. Hence u_X is a monomorphism by proposition 2.11 c). Now given two morphisms of simplicial sets $f, g : X \to Y$ such that $|f| = |g|$. Then $u_Y \circ f = |f|_\# \circ u_X = |g|_\# \circ u_X = u_Y \circ g$. So, because u_X is a monomorphism, we conclude $f = g$, i.e. $|\cdot| : \mathbf{sSet} \to \mathbf{Top}$ is faithful. Faithful functors reflect both epi- and monomorphisms, so the proposition follows from remark 2.1. □

Remark 2.27. Given a monomorphisms $f : X \to Y$ of simplicial sets. Then f factors as $X \to f(X) \hookrightarrow Y$, where the first map is an isomorphism and $|f(X)|^{\mathbf{CW}}$ is a subcomplex of $|Y|^{\mathbf{CW}}$. In particular, $|f|$ is a closed embedding.

2.5. Discrete simplicial sets

Construction 2.28 (The constant simplicial object functor)**.** For each set F we define the *constant simplicial set with value F*, denoted $\underline{F}$, to be the simplicial set with $\underline{F}_n = F$ for all $n \geq 0$ and $d_i : \underline{F}_{n+1} \to \underline{F}_n, s_i : \underline{F}_n \to \underline{F}_{n+1}$ the identity for all $i \in [n]$. We define the *constant simplicial object functor* $\underline{\cdot} : \mathbf{Set} \to \mathbf{sSet}$ by assigning to each set F the constant simplicial set $\underline{F}$ and assigning to any map $g : F \to G$ the unique map $\underline{g} : \underline{F} \to \underline{G}$ with $\underline{g}_0 = g$ (This forces $\underline{g}_n = g$ for all $n \geq 0$ by the simplicial identities).

Definition 2.29 (The evaluation functor)**.** For each nonnegative integer n we define the n^{th}-*evaluation functor* by

$$\mathrm{ev}_n : \mathbf{sSet} \to \mathbf{Set}, \quad X \mapsto X_n, \quad \text{and} \quad (f : X \to Y) \mapsto (f_n : X_n \to Y_n)$$

Further, we let $\mathbf{sSet}_{\leq n}$ denote the full subcategeory of $\mathbf{sSet}$ consisting of simplicial sets of dimension $\leq n$. A simplicial set of dimension ≤ 0 will be called *discrete*.

Proposition 2.30. *Given a simplicial complex X and a set F. Then the map*

$$\eta_{F,X} : Hom_{\boldsymbol{sSet}}(\underline{F}, X) \to Hom_{\boldsymbol{Set}}(F, ev_0(X)),\ f \mapsto ev_0(f)$$

is a bijection and $\eta_{F,X}$ is natural in F and X. In particular, $\underline{\cdot}$ is left adjoint to ev_0. Further, the functor $\underline{\cdot} : \boldsymbol{Set} \to \boldsymbol{sSet}$ is fully faithful and has essential image $\boldsymbol{sSet}_{\leq 0}$. Moreover, $ev_0 \circ \underline{\cdot} = id_{\boldsymbol{Set}}$ and the composition $\underline{\cdot} \circ ev_0|_{\boldsymbol{sSet}_{\leq 0}}$ is naturally isomorphic to $id_{\boldsymbol{sSet}_{\leq 0}}$. In particular, ev_0 restricts to an equivalence of categories $\boldsymbol{sSet}_{\leq 0} \to \boldsymbol{Set}$.

Proof. See [10, Subsection 00FQ]. □

2.6. Connectivity

Definition 2.31 (Summand, connectivity, connected component)**.** Given a simplicial set X. We say that a simplicial subset Y of X is a *summand* of X if there exists another simplicial subset Y' of X such that the canonical map $Y \coprod Y' \to X$ is an isomorphism. We say that X is *connected* if X is nonempty and every summand $Y \subseteq X$ is either empty or coincides with X. A simplicial subset Y of X is called a *connected component* of X if Y is a summand of X and Y is a connected simplicial set.

Remark 2.32 (Alternative characterization of summands)*.* Let X be a simplicial set and Y a simplicial subset of X. If Y is a summand of X, then there exists a unique simplicial subset Y' of X such that the canonical map $Y \coprod Y' \to X$ is an isomorphism: For each $n \geq 0$ we have $Y'_n = X_n \setminus Y_n$. Consequently, Y is a summand of X if and only if the face and degeneracy maps of X preserve the subsets $X_n \setminus Y_n$.

Example 2.33. Given a set I, then set of connected components of the simplicial set $\underline{I}$ is given by $\{\underline{\{i\}} \subseteq \underline{I}\}_{i \in I}$.

Remark 2.34*.* It follows from remark 2.32 that collection of summands of a fixed simplicial set X is closed under intersections and unions. Further, if Y is a summand of X and Y' is a summand of Y, then Y' is a summand of X.

Remark 2.35*.* Given a morphism of simplicial sets $f : X \to Y$ and simplicial subsets $Y' \subseteq Y$, $X' \subseteq X$. If Y' is a summand of Y then $f^{-1}(Y)$ is a summand of X. If X' is connected, then $f(X')$ is connected.

Proposition 2.36. *[10, Proposition 00GG] Let $f : X \to Y$ a morphism of simplicial sets and $X' \subseteq Y$ a simplicial subset of X, which is connected. Then there exists a unique connected component Y' of Y such that $f(X) \subseteq Y'$.*

The next assertion follows from proposition 2.36 because Δ^n defines a connected simplicial set for all $n \geq 0$.

Proposition 2.37. *For any simplicial set X the canonical map from the disjoint union of all connected components of X to X is an isomorphism.*

Definition 2.38 (Component map)**.** Given a simplicial set X and a set I. A morphism $\nu_X : X \to \underline{I}$ is said to be a *component map of X*, if for every set J

$$\mathrm{Hom}_{\mathbf{Set}}(I, J) \to \mathrm{Hom}_{\mathbf{sSet}}(\underline{I}, \underline{J}) \xrightarrow{\nu_X^*} \mathrm{Hom}_{\mathbf{sSet}}(X, \underline{J})$$

is a bijection, where ν_X^* denotes pre-composition by ν_X and the first map is the application of the constant simplicial object functor.

Remark. By proposition 2.30 the constant simplical object functor is fully-faithful, so, in situation of definition 2.38 ν_X is component map if and only if precomposition with ν_x defines a bijection $\mathrm{Hom}_{\mathbf{sSet}}(\underline{I}, \underline{J}) \xrightarrow{\nu_X^*} \mathrm{Hom}_{\mathbf{sSet}}(X, \underline{J})$ for all sets J.

Construction 2.39 (The functor π_0)**.** For any simplicial set X define $\pi_0(X)$ as the set of connected components of X. By proposition 2.36 for every n-simplex σ of X there exists a unique component $X' \subseteq X$ of X which contains σ. The construction $\sigma \to X'$ determines a map $\nu_X : X \to \underline{\pi_0(X)}$ and this map will be a component map of X, see [10, Proposition 00GQ]. So, by the universal property characterization of adjunction [12, Prop. 2.10], π_0 is turned into a left-adjoint of the constant simplicial set functor by declaring $\pi_0(f)$ to be unique map with $\underline{\pi_0(f)} \circ \nu_X = \nu_Y \circ f$ for every $f : X \to Y$.

The following two facts are now straightforward to check.

Remark. Let X, Y be simplicial sets. Firstly, For $f : X \to Y$ a morphism of simplicial sets $\pi_0(f)$ sends a connected component of X to the unique connected component of Y that contains X. Secondly, the map, which sends a component of X to the set of its connected components, determines a bijection from the set of components of X to the power set of the set of connected components of X.

Definition 2.40 (The category of directed graphs)**.** A *directed graph* G is a tuple $(\mathrm{Vert}(G), \mathrm{Edges}(G), s, t)$, where $\mathrm{Vert}(G)$ and $\mathrm{Edges}(G)$ are sets, whose elements are refered to as *vertices* and *edges*, respectively. $s, t : \mathrm{Edges}(G) \to \mathrm{Vert}(G)$ are functions assigning to each edge $e \in \mathrm{Edges}(G)$ its *source* $s(e) \in \mathrm{Vert}(G)$ and *target* $t(e) \in \mathrm{Edges}(G)$. For two directed graphs G, G' a morphism $G \to G'$ is a function $\mathrm{Edges}(G) \sqcup \mathrm{Vert}(G) \to \mathrm{Edges}(G') \sqcup \mathrm{Vert}(G')$ satisfying

(I) $f(v) \in \mathrm{Vert}(G')$ for all $v \in \mathrm{Vert}(G)$.

(II) For $e \in \mathrm{Edges}(G)$, if $f(e) \in \mathrm{Edges}(G)$, then $s(f(e)) = f(s(e))$ and $t(f(e)) = f(t(e))$, else $f(e) = f(s(e)) = f(t(e))$.

The category of directed graphs will be denoted by **Graph**.

Construction 2.41 (The Graph functor)**.** To every simplicial set X we assign a graph $\mathrm{Graph}(X)$ by setting $\mathrm{Vert}(\mathrm{Graph}(X)) = s_0(X_0)$, $\mathrm{Edges}(\mathrm{Graph}(X)) = X_1^{\mathrm{nd}}$, $s = s_0 \circ d_1$ and $t = s_0 \circ d_0$. Then $X_1 = \mathrm{Vert}(\mathrm{Graph}(X)) \sqcup \mathrm{Edges}(\mathrm{Graph}(X))$ and for every morphism of simplicial sets $f : X \to Y$ we let $\mathrm{Graph}(f)$ be the morphism $\mathrm{Graph}(X) \to \mathrm{Graph}(Y)$ given by $f_1 : X_1 \to Y_1$.

Proposition 2.42. *[10, Proposition 001N] The functor Graph* : ***sSet*** $\to$ ***Graph*** *induces an equivalence of categories from the full subcategory* $\boldsymbol{sSet}_{\leq 1} \subset \boldsymbol{sSet}$ *of simplicial sets of dimension* ≤ 1 *to the category* ***Graph****.*

Construction 2.43 (π_0 factors through the graph functor)**.** [10, Variant 00GV] Let $\pi_0(\mathrm{Graph}(X))$ denote the components of $\mathrm{Graph}(X)$. Given $\sigma \in X_n$. Then there exists a unique $F(\sigma) \in \pi_0(\mathrm{Graph}(X))$ with $\sigma(0) \in \mathrm{Vert}(F(\sigma))$ and then automatically $\sigma(i) \in \mathrm{Vert}(F(\sigma))$ for all $i \in [n]$. Hence, the construction $\sigma \mapsto F(\sigma)$ induces a map of simplicial sets $X \to \underline{\pi_0(\mathrm{Graph}(X))}$. This map is a component map of X.

Proposition 2.44 (π_0 factors through geometric realization)**.** *[10, Corollary 00H6] Given a simplicial set X. $Y' \subseteq Y$ is a summand of X if and only if $|Y'|$ is a summand of $|X|$. Let us denote by π_0 :* ***Top*** $\to$ ***Set*** *the functor that assign to a topological space is path-components. Then the map $X \to \underline{\pi_0(|X|)}$ sending a simplex $\sigma \in X_n$ to the path-component containing $|\sigma|$ is a component map of X.*

2.7. Homotopy and homology groups of simplicial sets

Definition 2.45 (Tuples of simplicial sets and slice categories)**.** Let k be a nonnegative integer, X a simplicial set and S a topological space.

1. We denote by $\mathbf{sSet}/X$ the slice category of $\mathbf{sSet}$ over X and similarly by $\mathbf{Top}/S$ the category of spaces over S. The morphisms in these categories are called *fiber preserving.*

2. We define $\mathbf{sSet}(k)$ as the category whose objects are k-tuples of simplicial sets $(X_1, \dots, X_k)$ together with monomorphisms $X_k \hookrightarrow X_{k-1} \hookrightarrow \dots \hookrightarrow X_2 \hookrightarrow X_1$. The morphisms in $\mathbf{sSet}(k)$ are given by *filtration preserving maps.*

3. We define $\mathbf{sSet}^*(k)$ as the full subcategory of $\mathbf{sSet}(k+1)$ consisting of pairs $(X_1, \dots, X_{k+1})$ with X_{k+1} a vertex of X, called the *base point.*

Clearly, we can make analogous definitions for $\mathbf{Top}(k), \mathbf{CW}(k)$ and $\mathbf{Top}^*(k), \mathbf{CW}^*(k)$ by replacing the word monomorphism with embedding/inclusion of a subcomplex, respectively. We denote $\mathbf{sSet}(1) = \mathbf{sSet}$ and $\mathbf{sSet}^* = \mathbf{sSet}^*(1)$.

Proposition/Definition 2.46. *Given a simplicial set X, we define the chain complex $C^{\Delta}(X) \in \boldsymbol{Ch}(\boldsymbol{Ab})$ and the normalized chain complex $C^{\boldsymbol{CW}}(X) \in \boldsymbol{Ch}(\boldsymbol{Ab})$ by definining $C^{\Delta}_n(X) = \mathbb{Z}\{X_n\}$ and $C^{\boldsymbol{CW}}_n(X) = \mathbb{Z}\{X^{nd}_n\}$ for all $n \geq 0$ and $C^{\Delta}_n(X) = C^{\boldsymbol{CW}}_n(X) = 0$ for $n < 0$. For $n > 0$ and $\sigma \in X_n$ and $\tau \in X^{nd}_n$ we define*

$$\partial_n(\sigma) = \sum_{i=0}^{n}(-1)^i d_i(\sigma) \text{ and } \partial_n(\tau) = \sum_{i=0}^{n} d_i(\sigma) \cdot \begin{cases} 0 & \text{if } d_i(\sigma) \text{ is degenerate,} \\ (-1)^i & \text{else.} \end{cases}$$

For $(X, A) \in \boldsymbol{sSet}(2)$ we set $C^{\Delta}(X, A) = C^{\Delta}(X)/C^{\Delta}(A)$ and $C^{\boldsymbol{CW}} = C^{\boldsymbol{CW}}/C^{\boldsymbol{CW}}(A)$. Further, we define $C(X, A) = C^{\Delta}(Sing(|X|), Sing(|A|))$.

In this way we obtain three functors $C^{\Delta}, C^{\boldsymbol{CW}}, C : \boldsymbol{sSet}(2) \to \boldsymbol{Ch}(\boldsymbol{Ab})$ In the first to cases, we assign to $f : (X, A) \to (Y, B)$ the chain map $f_{\#}$ that on simplices is given by $f_{\#}(\sigma) = f(\sigma)$. We view the third functor, as a composition of functors.

Proof. That these are indeed chain complexes follows from the simplicial identity (1). □

Proposition/Definition 2.47. *[8, Theorem 2.35] Given a pair of simplicial sets (X, A). Then $C^{\boldsymbol{CW}}(|X|^{\boldsymbol{CW}}, |A|^{\boldsymbol{CW}}) = C^{\boldsymbol{CW}}(X, A)$ and $C(|X|, |A|) = C(X, A)$. Further, the canonical natural chain map $C^{\Delta}(X, A) \hookrightarrow C(X, A)$ induce an isomorphism on homology. The canonical natural map of graded abelian groups*

$$H_*(C^{\boldsymbol{CW}}(X, A)) \to H_*(C^{\Delta}(X, A)),\ [\sigma] \mapsto [\sigma]$$

is an isomorphism. Let $\boldsymbol{Ab}_{\bullet}$ denote the category of graded abelian groups. We obtain functors $H_* : \boldsymbol{sSet}(2) \to \boldsymbol{Ab}_{\bullet}$ by composing $C, C^{\boldsymbol{CW}}$ or C^{Δ} with the homology functor $\boldsymbol{Ch}(\boldsymbol{Ab}) \to \boldsymbol{Ab}_{\bullet}$. These three functors are naturally isomorphic and are each referred to as the homology functor.

Definition 2.48 (Homotopy groups)**.** We define the *homotopy group* functors as the following compositions

$$\pi_1 : \mathbf{sSet}^*(k) \xrightarrow{|\cdot|} \mathbf{Top}^*(k) \xrightarrow{\pi_1} \mathbf{Grp} \quad \text{and} \quad \pi_n : \mathbf{sSet}^*(k) \xrightarrow{|\cdot|} \mathbf{Top}^*(k) \xrightarrow{\pi_n} \mathbf{Ab}$$

for any $n \geq 2$ and $k = 1, 2$.

For any set M let $F(M)$ denote the free group on M. Throughout this whole bachelor thesis, we will denote the unit of any group as 1.

Construction/Proposition 2.49 (Intrinsic discription of π_1)**.** *Given $Y \in$ **sSet*** with base point $v \in Y_0$. Let X be the connected component of Y containing v. Let $\Gamma \in$ **Graph** be a spanning tree in Graph(X). For any $e \in$ Edges(Graph$(X)) \setminus$ Edges(Γ) choose a loop γ_e in $|X|$ based at v, which is contained in $|\Gamma| \cup |e|$ and traverses $|e|$ exactly once and from $|e(0)|$ to $|e(1)|$. Then*

$$a : F(\mathit{Edges}(\mathit{Graph}(X)) \setminus \mathit{Edges}(\Gamma)) \to \pi_1(Y, x), \quad e \mapsto [\gamma_e]$$

is surjective with kernel the normal closure of the image of

$$b : F(X_2^{nd}) \to F(X_1) \to F(\mathit{Edges}(\mathit{Graph}(X)) \setminus \mathit{Edges}(\Gamma)),$$

where the left map is defined by $\sigma \mapsto d_0(\sigma)d_1(\sigma)^{-1}d_2(\sigma)$ and the right map by

$$e \mapsto \begin{cases} 1, & \text{if } e \text{ is degenerate or } e \in \mathit{Edges}(\Gamma) \\ e, & \text{else.} \end{cases}$$

Proof. Because attaching cells of dimension ≥ 3 has no effects on the fundamental group, we may assume that X is two dimensional. Further, we might assume that Edges$(\Gamma) = \emptyset$ by collapsing the contractible subcomplex $|\Gamma|$ of $|X|^{\mathbf{CW}}$ to $|v|$. Now, since $|\Gamma| = |v|$ is a spanning tree in $\mathrm{sk}_1(|X|^{\mathbf{CW}})$, the CW-complex $|X|^{\mathbf{CW}}$ is given by a wedge of circles, one for each $e \in$ Edges(Graph(X)), together with a collection of two-dimensional cells being attached to that wedge. Since two-dimensional cells are simply-connected every path in $|X|$ is homotopic to a path contained in $|\mathrm{sk}_1(X)|$. The map a is an isomorphism, when $X_2^{\mathrm{nd}} = \emptyset$ by the Seifert-van-Kampen theorem, so surjectivity is proven. By simply-connectedness of two-dimensional cells, a path that traverses the boundary of a two-cell is trivial in the fundamental group of $|X|$, so the kernel of a is at least as big as what we are claiming. Now for the converse, let $e \in$ Edges(Graph(X)) with γ_e nullhomotopic. The image of the corresponding homotopy contains a finite number of 2-cells of $|X|^{\mathbf{CW}}$. Hence we might assume that X is finite. We proceed by induction on the elements of X_2^{nd}, where the case $X_2^{\mathrm{nd}} = \emptyset$ is already settled. But, by [9, Prop. 10.13], the effect of attaching another 2-simplex $\sigma \in X_n^{\mathrm{nd}}$ is exactly adding the relation $b(\sigma)$ to a presentation of $\pi(|X|)$, which proves the proposition. □

3. Covering Space Projections and Simplicial Sets

Proposition/Definition 3.1. *[10, Proposition 021M] Let $p : Z \to X$ be a morphism of simplicial sets. Then the following are equivalent:*

a) *For every $n \geq 0$ and $\sigma : \Delta^n \to X$ the canonical map $p' : \Delta^n \times_X Z \to \Delta^n$ restricts to an isomorphism on each connected component.*

b) *Given $n, m \geq 0$ and $\alpha : \Delta^m \to \Delta^n$. For any $\sigma : \Delta^n \to X$ and $\tilde{\tau} : \Delta^m \to Z$ with $p \circ \tilde{\tau} = \sigma \circ \alpha$ there is a unique $\tilde{\sigma} : \Delta^n \to Z$ with $\tilde{\sigma} \circ \alpha = \tilde{\tau}$ and $p(\tilde{\sigma}) = \sigma$.*

If any of these conditions apply, we call p a (simplicial) covering projection or (simplicial) covering (space) map.

Proof. We assume a) and prove b). Given $\alpha : \Delta^m \to \Delta^n$, as well as $\sigma : \Delta^n \to X$ and $\tilde{\tau} : \Delta^m \to Z$ with $p(\tilde{\tau}) = \sigma \circ \alpha$. Let us denote the canonical map $\Delta^n \times_X Z \to Z$ by f. By the universal property of the pullback there is a unique $\tau' : \Delta^m \to \Delta^n \times_X Z$ such that $f(\tau') = \tilde{\tau}$ and $p'(\tau') = \alpha$. Let us denote the inverse map of the restriction of p' to the connected component that contains τ' by $p^- : \Delta^n \to \Delta^n \times_X Z$. Now given any $\rho : \Delta^n \to Z$ with $p(\rho) = \sigma$ and $\rho \circ \alpha = \tilde{\tau}$. Then there exists a unique map $\rho' : \Delta^n \to \Delta^n \times_X Z$ such that $p' \circ \rho' = \mathrm{id}_{\Delta^n}$ and $f \circ \rho' = \rho$. Since $f \circ \rho' \circ \alpha = \tilde{\tau}$ and $p' \circ \rho' \circ \alpha = \alpha$, we have $\rho' \circ \alpha = \tau'$. We see that both $p^-, \rho' : \Delta^n \to \Delta^n \times_X \Delta^n$ map into the same connected component of $\Delta^n \times_X Z$ and by uniqueness of inverses, we conclude $\rho' = p^-$. Therefore $\rho = f \circ \rho' = f \circ p^- =: \tilde{\sigma}$. Since $p \circ f \circ p^- = \sigma \circ p' \circ p^- = \sigma$ and $f \circ p^- \circ \alpha = f \circ \tau' = \tilde{\tau}$, the unique map such that $p(\tilde{\sigma}) = \sigma$ and $\tilde{\sigma} \circ \alpha = \tilde{\tau}$ is $f \circ \rho^- = \tilde{\sigma} : \Delta^n \to Z$.
Conversely, assume b) and let us prove a). Given $\sigma : \Delta^n \to X$. Let $F := Z_n \times_{X_n} \{\sigma\}$ denote the collection of all n-simplices of $\tilde{\sigma}$ of Z with $p(\tilde{\sigma}) = \sigma$. It suffices to show that the map $h : \coprod_{\tilde{\sigma} \in F} \Delta^n \to \Delta^n \times_X Z$, which on every component is given by $(\mathrm{id}_{\Delta^n}, \tilde{\sigma})$, is an isomorphism of simplicial sets. We can check this levelwise, i.e. let $m \geq 0$ and we have to show h_m is bijection. An m-simplex ρ of $\Delta^n \times_X Z$ is an m-simplex of Δ^n, say $\alpha : \Delta^m \to \Delta^n$, and a m-simplex of Z, say $\tilde{\tau} : \Delta^m \to Z$, such that $p \circ \tilde{\tau} = \sigma \circ \alpha$. An m-simplex $\tilde{\rho}$ of $\coprod_{\tilde{\sigma} \in T} \Delta^n$ is given by an m-simplex of Δ^n, say $\beta : \Delta^m \to \Delta^n$, and $\tilde{\sigma}$ with $p(\tilde{\sigma}) = \sigma$. Now $h_m(\tilde{\rho}) = \rho$ if and only if $\alpha = \beta$ and $\tilde{\sigma} \circ \alpha = \tilde{\tau}$. So b) says that every m-simplex of $\Delta^m \times_X Z$ has a unique preimage under h_m. □

3.1. Simplicial vs topological covering projections

Our goal now is to compare this definition of simplicial covering projection to the theory of topological covering spaces.

Definition 3.2 (Topological covering space map). Let S, T be topological spaces.

a) Given a continuous map $p : S \to T$. A subspace $U \subseteq T$ is called *evenly covered* by p if $p|_{p^{-1}(U)} : p^{-1}(U) \to U$ restricts to a homeomorphism on every connected component of $p^{-1}(U)$. Note that U is evenly covered by p if and only if there exists a discrete topological space F and a homeomorphism $h : U \times F \to p^{-1}(U)$ such that

$$\begin{tikzcd} U \times F \arrow[r, "h"] \arrow[dr, "\mathrm{pr}_U"'] & p^{-1}(U) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner"] & S \arrow[d, "p"] \\ & U \arrow[r, hook] & T \end{tikzcd}$$

commutes.

b) A continuous map $p : S \to T$ of topological spaces is called *covering (space) map*, or *a covering (space) projection onto* T, if for any $t \in T$ there exists an open neighborhood $U \subseteq T$ of t such that U is *evenly covered* by p.

c) A continuous map $p : S \to T$ is called a *trivial covering space projection*, if T is evenly covered by p.

d) For $p : S \to T$ a covering space map and $f \in \mathbf{Top}/T$. A morphism $f \to p$ in $\mathbf{Top}/T$ is called *a lift of f along p.*

e) The full subcategory of $\mathbf{Top}/T$ consisting of covering space projections onto T is denoted by $\mathbf{Cov}(T)$.

f) For a continuous map $p : S \to T$ the group of homeomorphisms of S that lift p along p is denoted $\mathrm{Aut}(p) := \mathrm{Aut}_{\mathbf{Top}/T}(p)$. If p is a covering space projection, the elements of $\mathrm{Aut}(p)$ are called *deck transformations.* In that case, p is called *normal* if $\mathrm{Aut}(p)$ acts transitively on $p^{-1}(\{t\})$ for all $t \in T$.

Proposition 3.3. *[8, sec. 1.3] Let S, T, S' be topological spaces with $s_0 \in S, t_0 \in t$ and $s'_0 \in S'$ points.*

a*) (Uniqueness of lifts:) Given a covering map $p : S \to T$ and a continuous map $f : S' \to T$. If S' is connected than two lifts of f agree if they agree on a point.*

b*) (Lifting criterion:) Suppose $p : (S, s_0) \to (T, t_0)$ is a covering map. For any $f : (S', s'_0) \to (T, t_0)$ with S' connected and locally path-connected, there exists a lift $g : (S', s'_0) \to (S, s_0)$ of f if and only if $f_*(\pi_1(S', s'_0)) \subseteq p_*(\pi_1(S, s_0))$.*

c*) Suppose $p : (S, s_0) \to (T, t_0)$ is a normal covering map. For $\gamma : (I, \partial I) \to (T, t_0)$ there is an unique $g \in Aut(p)$ such that the unique $\tilde{\gamma} : (I, 0) \to (S, s_0)$ lifting γ along p has endpoint $g(s_0)$. The assignment $[\gamma] \mapsto g$ gives a well-defined homomorphism $L_{p,s_0} : \pi_1(X, t_0) \to Aut(p)$. L_{p,s_0} has kernel $im(\pi_1(p, s_0))$. When S is connected, then L_{p,s_0} is surjective.*

Corollary 3.4. *Let S, T be topological spaces and $p : S \to T$ a covering space projection. If T is simply-connected and locally path-connected then p is trivial.*

Proof. We assume that S is connected, in particular, $S \neq \emptyset$, and have to prove that p is a homeomorphism. So, let $s_0 \in S$ and $t_0 := p(s_0)$. Applying the lifting criterion 3.3 b) to the covering space map $p : (S, s_0) \to (T, t_0)$ and the continuous map $\mathrm{id}_T : (T, t_0) \to (T, t_0)$ we obtain a continuous map $g : (T, t_0) \to (S, s_0)$ with $p \circ g = \mathrm{id}_T$. Now $g \circ p : S \to S$ lifts $p : S \to T$ along $p : S \to T$ and has $g \circ p(s_0) = s_0$. The identity $\mathrm{id} : S \to S$ has the same properties, so by proposition 3.3a) we have $g \circ p = \mathrm{id}_S$. p is surjective because the cardinality of the fiber $p^{-1}(x)$ is constant on any path-component of T. Therefore, $g = p^{-1}$. □

Lemma 3.5. *Let $p : S \to T$ and $f : T' \to T$ be continuous maps of topological spaces. Let $p' : S' := S \times_T T' \to T'$, $f' : S' \to S$ denote the canonical maps. Let $B_{p,f}$ denote the group homomorphism $Aut(p) \to Aut(p'), g \mapsto g \times_T id_{T'}$. If $U \subseteq T$ is evenly covered by p, then $f^{-1}(U) \subseteq T'$ is evenly covered by p'. In particular, if p is a (trivial) covering space projection, then so is p'.*
From now on assume that p is a covering space projection and that there exists basepoints $s_0' \in S'$, $s_0 := f'(s_0'), t_0' := p'(s_0'), t_0 = p(s_0)$. Then $\pi_1(p', s_0')$ is an isomorphism onto

$$\pi_1(f, t_0')^{-1}(im\ \pi_1(p, s_0))) \subseteq \pi_1(T', t_0').$$

If p is normal, then so is p', and $L_{p',s_0'} = B_{p,f} \circ L_{p,s_0} \circ \pi_1(f, t_0')$. When S' is path-connected, then

$$(im\ \pi_1(f, t_0')) \cdot (im\ \pi_1(p, s_0)) = \pi_1(T, t_0),$$

and the converse holds, when T' is path-connected. If S is connected, then $B_{p,f}$ is injective. If S' is connected and p is normal, then $B_{p,f}$ is surjective.

Proof. Suppose U is evenly covered by p. Choose an open neighborhood $U \subseteq S$ of $p(t')$ that is evenly covered by p. Let F be a discrete topological space and $h : U \times F \to p^{-1}(U)$ a homeomorphism such that

$$\begin{tikzcd}
U \times F \arrow[r, "h"] \arrow[dr, "\mathrm{pr}_U"'] & p^{-1}(U) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d, "p"] \\
& U \arrow[r, hook] & T
\end{tikzcd}$$

commutes. We denote $V := f^{-1}(U)$ and first only consider the solid arrows in the following cube:

$$\begin{tikzcd}
& p^{-1}(U) \arrow[rr, hook] \arrow[d] & & S \arrow[d, "p"] \\
& U \arrow[rr, hook] & & T \\
p'^{-1}(V) \arrow[ruu, dashed, "f'|_{p'^{-1}(V)}"] \arrow[rr, hook] \arrow[d] & & S' \arrow[ruu, "f'"'] \arrow[d, "p'"'] & \\
V \arrow[ruu, "f|_V"] \arrow[rr, hook] & & T' \arrow[ruu, "f"'] &
\end{tikzcd}$$

Note that the four squares with solid boundaries in this diagram are pullback squares and that the solid diagram commutes. By the universal property of the back-face pullback square, we can fill in a unique dashed arrow such that the whole cube commutes. By the pasting law for pullbacks, see [16], the left-side face of the commutative cube above is a pullback square. So, applying its universal property we obtain the following commutative diagram:

$$\begin{tikzcd}
F \times V \arrow[r, "\mathrm{id}_F \times f|_V"] \arrow[d, dashed, "\exists!"] \arrow[dd, bend right=60, "\mathrm{pr}_V"'] & F \times U \arrow[d, "h"'] \arrow[dd, bend left=60, "\mathrm{pr}_U"] \\
p'^{-1}(V) \arrow[r, "f'|_{p'^{-1}(V)}"'] \arrow[d] & p^{-1}(U) \arrow[d] \\
V \arrow[r, "f|_V"] & U
\end{tikzcd}$$

Its straightforward to check that in this diagram the outer square is a pullback square. So, again by the pasting law for pullbacks, the upper square is a pullback square, too. Hence the map $F \times V \to p'^{-1}(V)$ is the pullback of a homeomorphism and therefore itself a homeomorphism by [16], i.e. V is evenly covered by p'.

Throughout all remaining parts of the proof we will assume that p is a covering space projection.

Since p' is a covering map, $\pi_1(p', s'_0)$ is injective. Because $\pi_1(f) \circ \pi_1(p') = \pi_1(p) \circ \pi_1(f')$, the image of $\pi_1(p', s'_0)$ is contained in $\pi_1(f, t_0)^{-1}(\mathrm{im}\ \pi_1(p, s_0)))$. To see the reverse inclusion, let $\gamma : (I, \partial I) \to (T', t'_0)$ be a loop such that there exists a path $\tilde{\gamma} : (I, \partial I) \to (S, s_0)$ with $f \circ \gamma = p \circ \tilde{\gamma}$. By proposition 3.3 b) there exists a unique map $\tilde{\gamma}' : (I, 0) \to (S', s'_0)$ with $p' \circ \tilde{\gamma}' = \gamma$. Both paths $f'\tilde{\gamma}'$ and $\tilde{\gamma}$ start at s_0 and lift $f\gamma$. So, by 3.3 a), $f'\tilde{\gamma}'(1) = \tilde{\gamma}(1) = s_0$. Since $p'(\tilde{\gamma}'(1)) = \gamma(1) = t'_0 = p'(s'_0)$ and $f'(\tilde{\gamma}'(1)) = s_0 = f'(s'_0)$ we obtain $\tilde{\gamma}'(1) = s'_0$ by the universal property of the pullback. Therefore $[\tilde{\gamma}] \in \pi_1(S', s'_0)$ is a preimage of γ under $\pi_1(p')$.

Suppose p is normal and let $s', s'' \in S'$ with $p'(s') = p'(s'')$. Then $pf'(s') = pf'(s'')$ and therefore there exists $g \in \mathrm{Aut}(p)$ with $f'(s') = gf'(s'')$. Let $g' := g \times_T \mathrm{id}'_T \in \mathrm{Aut}(p')$. Then $p(g'(s'')) = p(s'') = p(s')$ and $f'(g'(s'')) = gf'(s'') = f'(s')$. So by the universal property of the pullback $g'(s'') = s'$, i.e. p' is normal.

Again suppose p is normal and take $\gamma' : (I, \partial I) \to (T', t'_0)$. Let $\tilde{\gamma}' : (I, 0) \to (S', s'_0)$ be a lift of γ' along p'. Further, let $\tilde{\gamma} : (I, 0) \to (S, s_0)$ be a lift of $f \circ \gamma'$ along p. Let $g \in \mathrm{Aut}(p)$ with $\tilde{\gamma}(1) = g(s_0)$ and define $g' := B_{p,f} = g \times_T \mathrm{id}_{T'} \in \mathrm{Aut}(p')$. Since both paths $f' \circ \tilde{\gamma}'$ and $\tilde{\gamma}$ lift $f \circ \gamma'$ and start at s_0, they have the same endpoint, namely $g(s_0)$, by proposition 3.3 a). The equations $f'(\tilde{\gamma}'(1)) = g(s_0) = f'(g'(s'_0))$ and $p'(\tilde{\gamma}'(1)) = t'_0 = p'(g'(s'_0))$ imply $\tilde{\gamma}'(1) = g'(s'_0)$ by the universal property of the pullback. So, unraveling the definitions

$$L_{p',s'_0}([\gamma']) = g' = B_{p,f}(g) = B_{p,f} \circ L_{p,s_0}([f \circ \gamma']) = B_{p,f} \circ L_{p,s_0} \circ \pi_1(f, t'_0)([\gamma']).$$

Now we suppose that T' is path-connected and that $(\mathrm{im}\pi_1(f, t'_0)) \cdot (\mathrm{im}\pi_1(p, s_0)) = \pi_1(T, t_0)$ holds. To prove that S' is path-connected, it suffices to find a path λ' :

$(I, 0, 1) \to (S', s_0', s_0'')$ for any $s_0'' \in S'$ with $p(s_0'') = t_0$, using that T' is path-connected and p' is a covering map. Let $\alpha : I \to S$ be a path from $f'(s_0') = s_0$ to $f'(s_0'')$. By assumption there exists $\lambda : (I, \partial I) \to (T', t_0')$ and $\beta : (I, \partial I) \to (S, s_0)$ such that $(f \circ \lambda) * (p \circ \beta)$ is path-homotopic to $p \circ \gamma$. Hence, $f \circ \lambda$ is path-homotopic to $p \circ (\gamma * \overline{\beta})$. By the path-lifting property 3.3 b) there exist $\tilde{\lambda} : (I, 0) \to (S, s_0)$ with $p \circ \tilde{\lambda} = f \circ \lambda$ and, by the homotopy lifting property 3.3 b), the homotopy $f \circ \lambda \simeq p \circ (\gamma * \overline{\beta})$ can be lifted to a homotopy $\tilde{\lambda} \simeq \gamma * \overline{\beta}$. Since the first homotopy fixes basepoints, the second must too, by proposition 3.3 a). Therefore, $\tilde{\lambda}(1) = (\gamma * \overline{\beta})(1) = f'(s_0'')$. By the universal property of the pullback there exists $\lambda' : I \to S'$ with $p \circ \lambda' = \lambda$ and $f' \circ \lambda' = \tilde{\lambda}$. Because $f'(\lambda'(0)) = s_0 = f'(s_0')$, $p'(\lambda'(0)) = t_0' = p'(s_0')$, we have $\lambda'(0) = s_0'$ by the universal property of the pullback. The same reasoning gives $\lambda'(1) = s_0''$. So, λ' is the required path and S' is path-connected.
Now let us assume that S' is path-connected and take any $\gamma : (I, \partial I) \to (T, t_0)$. By the path-lifting property 3.3 b) there exists $\tilde{\gamma} : (I, 0) \to (S, s_0)$ with $p\tilde{\gamma} = \gamma$. Since $p\tilde{\gamma}(1) = \gamma(1) = t_0 = f(t_0')$, there exists (a unique) $s_0'' \in S'$ with $f'(s_0'') = \tilde{\gamma}(1)$ and $p'(s_0'') = t_0'$. By assumption there exists a path $\alpha : (I, 0, 1) \to (S', s_0', s_0'')$. Define $\beta := \tilde{\gamma} * \overline{(f' \circ \alpha)} : (I, \partial I) \to (S, s_0)$. Then $\gamma = p \circ \tilde{\gamma} = p \circ (\beta * (f' \circ \alpha)) = (p \circ \beta) * (p \circ f' \circ \alpha) = (p \circ \beta) * (f \circ p' \circ \alpha)$, i.e. $[\gamma] = \pi_1(p, s_0)([\beta]) * \pi_1(f, t_0')([p' \circ \alpha])$, so, $(\mathrm{im}\ \pi_1(f, t_0')) \cdot (\mathrm{im}\ \pi_1(p, s_0)) = \pi_1(T, t_0)$.

Now suppose S is connected and let $g \in G$ with $\mathrm{id}_{T'} \times_T g' = \mathrm{id}_S'$. Then $g \circ f' = f' \circ (\mathrm{id}_{T'} \times_T g') = f'$, so g fixes s_0 and therefore agrees with id_S by proposition 3.3, i.e. $B_{p,f}$ is injective.
Suppose p is normal, S' is connected and let $g' \in \mathrm{Aut}(p')$. There exists $g \in \mathrm{Aut}(g)$ with $g(s_0) = f'(g'(s_0'))$. Both maps $f' \circ g'$ and $g \circ f'$ lift $f \circ p'$ along p and agree on s_0'. Hence, $f' \circ g' = g \circ f'$ by 3.3 a). But then $p' \circ (g \times_T \mathrm{id}_{T'}) = p = p' \circ g'$ and $f' \circ (g \times_T \mathrm{id}_{T'}) = g \circ f' = f' \circ g'$, so $(g \times_T \mathrm{id}_{T'}) = g'$ by the universal property of the pullback. This implies that $B_{p,f}$ is surjective. □

Construction 3.6 (Neighborhoods in the geometric realization)**.** Given a simplicial set X, a subset A of $|X|$ and a function $\varepsilon : \bigcup_{n \geq 0} X_n^{\mathrm{nd}} \to (0, \infty), \sigma \mapsto \varepsilon_\sigma$. We construct, by induction over the skeleta of X, an open neighborhood $N_\varepsilon(A) \subseteq |X|$ of A. Suppose we have already constructed an open neighborhood $N_\varepsilon^n(A)$ of $A \cap |\mathrm{sk}_n(X)|$ in $|\mathrm{sk}_n(X)|$, starting with $N_\varepsilon^0(A) = A \cap |\mathrm{sk}_0(X)|$. Then we define $N_\varepsilon^{n+1}(A)$ by specifying the preimage of $|\Phi_\sigma| : \Delta^{n+1} \to |\mathrm{sk}_{n+1}(X)|$ for all $\sigma \in X_{n+1}^{\mathrm{nd}}$. $|\Phi_\sigma|^{-1}(N_\varepsilon(A))$ shall be the union of two parts: An open ε_σ neighborhood of $|\Phi_\sigma|^{-1}(A) \setminus |\partial\Delta^{n+1}|$ in $|\Delta^{n+1}| \setminus |\partial\Delta^{n+1}|$ and

$$\{t \cdot G + (1-t) \cdot x \ : \ x \in |\sigma|^{-1}(N_\varepsilon^n(A)),\, t \in [0, \varepsilon_\sigma)\},$$

where G is the barycenter of Δ^{n+1}. Then we define $N_\varepsilon(A) := \bigcup_{n \geq 0} N_\varepsilon^n(A)$. This is an open set in $|X|$ because it preimage under each characteristic map of $|X|^{\mathbf{CW}}$ is open, further:

a) For $A, B \subseteq |X|$ disjoint closed sets, $N_\varepsilon(A)$ and $N_\varepsilon(B)$ are disjoint for small enough ε_σ's, see [8, Prop. A.3].

b) For $A \subseteq |X|$ a simplicial subset, $N_\varepsilon(A)$ strongly deformation retracts onto A if $\varepsilon_\sigma < 1$ for all $\sigma \in X_n^{\mathrm{nd}}$, $n > 0$, see [8, Prop. A5].

c) For $A \subseteq B \subseteq |X|$, we have $N_\varepsilon(A) \subseteq N_\varepsilon(B)$.

d) When $A, B \subseteq X$ are simplicial subsets, then $N_\varepsilon(A) \cap N_\varepsilon(B) = N_\varepsilon(A \cap B)$.

Lemma 3.7. *Given a morphism of simplicial sets $f : X \to Y$, a subset $A \subseteq X$ and functions $\varepsilon_X : \bigcup_{n\geq 0} X_n^{nd} \to (0,\infty), \sigma \mapsto \varepsilon_\sigma$ and $\varepsilon_Y : \bigcup_{n\geq 0} Y_n^{nd} \to (0,\infty), \sigma \mapsto \varepsilon_\sigma$ such that $\varepsilon_{f(\sigma)} = \varepsilon_\sigma$ for all $\sigma \in \bigcup_{n\geq 0} X_n^{nd}$. Then $|f|^{-1}(N_\varepsilon(A)) = N_\varepsilon(|f|^{-1}(A))$.*

Proof. For $\sigma \in X_n^{\mathrm{nd}}$, $|f| \circ |\sigma| = |f \circ \sigma|$ and so $|\sigma|^{-1}(|f|^{-1}((N_\varepsilon(A)))) = |f(\sigma)|^{-1}(N_\varepsilon(A))$, where the latter set has been specified inductively in construction 2.43 as exactly the same set as $|\sigma|^{-1}(N_\varepsilon(|f|^{-1}(A)))$. □

Theorem 3.8 (Topological vs. simplicial covering projections)**.** *[5, App. I. 3] For a morphism $p : Z \to X$ of simplicial sets the following are equivalent:*

a*) $|p| : |Z| \to |X|$ is a covering space projection by definition 3.2.*

b*) p is a simplicial covering space projection by definition 3.1.*

Proof. We assume that $|p|$ is a covering space map and prove part a) of definition 3.1. Let $n \geq 0$, $\sigma \in X_n$ and let $p' : \Delta^n \times_X Z \to \Delta^n$ be the canonical map. Since Δ^n is finite, we can conclude from remark 2.25 that $|p'|$ is the pullback of $|p|$ along $|\sigma|$. By lemma 3.5 $|p'|$ is a covering map. Since $|\Delta^n|$ is simply-connected and locally path-connected, $|p'|$ is a trivial covering space projection by corollary 3.4. So, $|p'|$ is a homeomorphism when restricted to any connected component. By 2.44, restriction to a connected component commutes with geometric realization. So, the statement follows because the geometric realization functor is conservative, as we have seen in 2.26.
Conversely, assume that b) holds and let us prove a). Let $x \in |X|$. Let n be the minimal nonnegative integer such that there exists $\sigma \in X_n^{\mathrm{nd}}$ with $x \in |\sigma|$, i.e. (n, σ) is the unique pair with $x \in |\sigma|(|\Delta^n| \setminus |\partial\Delta^n|)$.
Ad hoc definition: For any nondegenerate simplex $\sigma : \Delta^n \to X$ of a simplicial set, we define e_σ to be the open cell of $|X|^{\mathbf{CW}}$ corresponding to σ.
By proposition 3.1 a), there exists a fiber preserving isomorphism $h : \Delta^n \times F \to \Delta^n \times_X Z$, where F is a discrete simplicial set. Because F, Δ^n are locally finite, $|h| : |\Delta^n| \times |F| \to |\Delta^n| \times_{|X|} |Z|$ is a fiber-wise homeomorphism, using remark 2.25. Pulling $|h|$ back along the inclusion of $B := |\Delta^n| \setminus |\partial\Delta^n|$ into $|\Delta^n|$ yields that $|h| : B \times |F| \to B \times_{|X|} |Z|$ is a fiber-wise homeomorphism. But $|\sigma|$ restricted to B equals the inclusion of e_σ into $|X|$, therefore $B \times_{|X|} |Z|$ and $|p|^{-1}(e_\sigma)$ are fiber-wise homeomorphic. So, $|p|$ restricted to any connected component of $|p|^{-1}(e_\sigma)$ induces a homeomorphism onto e_σ.
Let $U := N_\varepsilon(e_\sigma)$ for a constant $0 < \varepsilon < 1$. By construction 3.6 U is an open neighborhood of x in $|X|$ and lemma 3.7 says that $|p|^{-1}(U) = N_\varepsilon(|p|^{-1}(e_\sigma))$ By construction 3.6 $|p|^{-1}(U)$ strongly deformation retracts onto $|p|^{-1}(e_\sigma)$. Let $\tilde\sigma : \Delta^n \to Z$ with $p(\tilde\sigma) = \sigma$. It remains to be shown that $|p|$ restricted to the unique connected component V of $|p|^{-1}(U)$

that contains $e_{\tilde{\sigma}}$ is an isomorphism. For another $\tilde{\sigma}_2 \in X_n$ with $p(\tilde{\sigma}_2) = \sigma$, we have that $N_\varepsilon(e_{\tilde{\sigma}}) \cap N_\varepsilon(e_{\tilde{\sigma}_2}) \neq \emptyset$ implies $\tilde{\sigma}_2 = \tilde{\sigma}$, using that $\varepsilon < 1$. Hence $V = N_\varepsilon(e_{\tilde{\sigma}})$. Inspection of the explicit formulas for $N_\varepsilon(e_\sigma)$ and $N_\varepsilon(e_{\tilde{\sigma}})$ tells us that it suffices to prove the following: For any $\tau \in X_m^{\text{nd}}$ with $d_i(\tau) = s_I(\sigma)$ for some $i \in [m]$ and I as in convention 2.12, there exists a unique $\tilde{\tau} \in Z_m^{\text{nd}}$ with $d_i(\tilde{\tau}) = s_I(\tilde{\sigma})$ and $p(\tilde{\tau}) = \tau$. But this proposition 3.1 b) applied to τ and $s_I(\tilde{\sigma})$ with $\alpha = \delta_i : \Delta^n \to \Delta^m$. □

Theorem 3.9. *Let X be a simplicial set, S a topological space and $f : S \to |X|$ be a covering space projection. Then there exists a simplicial set Z together with morphisms $p : Z \to X$ and $u_Z : Z \to Sing(S)$, such that u_Z exhibits S as a geometric realization of Z with $|f| = |p|$.*

Proof. We proceed by induction on skeleta. Let $Z^{(-1)}$ be the empty simplicial set and $p^{(-1)} : Z^{(-1)} \to \text{sk}_{-1}(X)$ and $u_{Z^{(-1)}} : Z^{(-1)} \to \text{Sing}(f^{-1}(|\text{sk}_{-1}(X)|))$ be empty maps. Then $u_{Z^{(-1)}}$ exhibits $\emptyset = f^{-1}(|\text{sk}_{-1}(X)|)$ as geometric realization of $Z^{(-1)}$ and $|p^{(-1)}| = f|_{f^{-1}(|\text{sk}_{-1}(X)|)} : f^{-1}(|\text{sk}_{-1}(X)|) \to |\text{sk}_{-1}(X)|$.
Given $n \geq 0$ and assume for all $-1 \leq m \leq n-1$ we have constructed simplicial sets $Z^{(m)}$ together with morphisms $p^{(m)} : Z^{(m)} \to \text{sk}_m(X)$, $u_{Z^{(m)}} : Z^{(m)} \to \text{Sing}(f^{-1}(|\text{sk}_m(X)|))$ such that

- $u_{Z^{(m)}}$ exhibits $f^{-1}(|\text{sk}_m(X)|)$ as geometric realization of $Z^{(m)}$
- $|p^{(m)}|$ equals $f|_{f^{-1}(|\text{sk}_m(X)|)}$ as continous map $f^{-1}(|\text{sk}_m(X)|) \to |\text{sk}_m(X)|$
- $\text{sk}_m(Z^{(n-1)}) = Z^{(m)}$ and $p^{(n-1)}|_{Z^{(m)}} = p^{(m)}$, as well as, $u_{Z^{(n-1)}}|_{Z^{(m)}} = u_{Z^{(m)}}$.

Let $\sigma : \Delta^n \to X$ be an n-simplex. Let $\phi_\sigma : \partial\Delta^n \to \text{sk}_{n-1}(X)$ be the attaching map and $\Phi_\sigma : \Delta^n \to \text{sk}_n(X)$ be the characteristic map of σ, as defined in 2.17. We construct the pullback of $p^{(n-1)}$ along ϕ_σ:

$$\begin{tikzcd}
\partial\Delta^n \times_{\text{sk}_{n-1}(X)} Z^{n-1} \arrow[r, "\phi'_\sigma"] \arrow[d, "(p^{(n-1)})'"'] \arrow[dr, phantom, "\lrcorner", very near start] & Z^{n-1} \arrow[d, "p^{n-1}"] \\
\partial\Delta^n \arrow[r, "\phi_\sigma"] & \text{sk}_{n-1}(X)
\end{tikzcd}$$

By lemma 3.5 and corollary 3.4, the pullback of f along $|\sigma|$ is a trivial covering space projection, using that $|\Delta^n|$ is simply-connected and locally path-connected. Concretely, there exists a set F_σ and family of continuous maps $(|\sigma|'_i : |\Delta^n| \to Z)_{i \in F_\sigma}$ such that

$$\begin{tikzcd}
\coprod_{i \in F_\sigma} |\Delta^n| \arrow[r, "(|\sigma|'_i)_{i \in I}"] \arrow[d, "(|\text{id}_{\Delta^n}|)_{i \in F_\sigma}"'] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d, "f"] \\
|\Delta^n| \arrow[r, "|\sigma|"] & |X|
\end{tikzcd}$$

is a pullback square. The outer square of the diagram

$$\begin{tikzcd}
\coprod_{i\in F_\sigma}|\partial\Delta^n| \arrow[r, hook] \arrow[d, "(|\mathrm{id}_{\partial\Delta^n}|)_{i\in F_\sigma}"] \arrow[dr, phantom, "\lrcorner", very near start] & \coprod_{i\in F_\sigma}|\Delta^n| \arrow[r, "(|\sigma|'_i)_{i\in I}"'] \arrow[d, "(|\mathrm{id}_{\Delta^n}|)_{i\in F_\sigma}"'] \arrow[dr, phantom, "\lrcorner", very near start] & Z \arrow[d, "f"'] \\
|\partial\Delta^n| \arrow[r, hook] & |\Delta^n| \arrow[r, "|\sigma|"] & |X|
\end{tikzcd}$$

and the outer square of the diagram

$$\begin{tikzcd}
|\partial\Delta^n\times_{\mathrm{sk}_{n-1}(X)}Z^{(n-1)}| \arrow[r, "|\phi'_\sigma|"] \arrow[d, "|(p^{(n-1)})'|"] \arrow[dr, phantom, "\lrcorner", very near start] & |Z^{(n-1)}| \arrow[r, hook] \arrow[d, "|p^{(n-1)}|"'] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d] \\
|\partial\Delta^n| \arrow[r, "|\phi_\sigma|"] & |\mathrm{sk}_{n-1}(X)| \arrow[r, hook] & |X|
\end{tikzcd}$$

are the pullback square of the same diagram using remark 2.25, the fact that $\partial\Delta^n$ is finite and the pasting law for pullbacks [16].
By proposition 2.44, restriction to connected components commutes with geometric realization and by proposition 2.26 geometric realization is conservative. So, because the same holds for $(|\mathrm{id}_{\partial\Delta^n}|_{i\in F_\sigma}) : \coprod_{i\in F_\sigma}|\partial\Delta^n| \to |\partial\Delta^n|$, $(p^{(n-1)})'$ is an isomorphism of simplicial sets, when restricted to any connected component of $\partial\Delta^n\times_{\mathrm{sk}_{n-1}(X)}Z^{(n-1)}$, where the set of connected components of $\partial\Delta^n\times_{\mathrm{sk}_{n-1}(X)}Z^{(n-1)}$ is in bijection to F_σ. Let $\rho_{i,\sigma} : \partial\Delta^n \to \partial\Delta^n\times_{\mathrm{sk}_{n-1}(X)}Z^{(n-1)}$ be the components of the isomorphism $\coprod_{i\in F_\sigma}\partial\Delta^n \to \partial\Delta^n\times_{\mathrm{sk}_{n-1}(X)}Z^{(n-1)}$ with $(p^{(n-1)})'\circ\rho_{i,\sigma} = \mathrm{id}_{\partial\Delta^n}$. Finally, we define $Z^{(n)}$ via the pushout

$$\begin{tikzcd}
\coprod_{\substack{\sigma\in X_n^{\mathrm{nd}}\\ i\in F_\sigma}}\partial\Delta^n \arrow[r, hook] \arrow[d, "(\phi'_\sigma\circ\rho_{i,\sigma})"'] & \coprod_{\substack{\sigma\in X_n^{\mathrm{nd}}\\ i\in F_\sigma}}\Delta^n \arrow[d, "(\tilde{\Phi}_{i,\sigma})"] \arrow[dl, phantom, "\ulcorner", very near end] \\
Z^{(n-1)} \arrow[r, hook] & Z^{(n)}
\end{tikzcd}$$

Further, let $p^{(n)} : Z^{(n)} \to \mathrm{sk}_n(X)$ be the unique map with $p^{(n)}\circ\tilde{\Phi}_{i,\sigma} = \Phi_\sigma$ for all $\sigma\in X_n^{\mathrm{nd}}$, $i\in F_\sigma$ and $p^{(n)}|_{Z^{(n-1)}} = p^{(n-1)}$. Let $u_{Z^{(n)}} : Z^{(n)} \to \mathrm{Sing}(f^{-1}(|\mathrm{sk}_n(X)|))$ be the unique map such that $u_{Z^{(n)}}|_{Z^{(n-1)}} = u_{Z^{(n-1)}}$ and such that for every $\sigma\in X_n^{\mathrm{nd}}$, $i\in F_\sigma$ the composition

$$\Delta^n \xrightarrow{\tilde{\Phi}_{i,\sigma}} Z^{(n)} \xrightarrow{u_{Z^{(n)}}} \mathrm{Sing}(f^{-1}(|\mathrm{sk}_n(X)|))$$

equals the n-simplex $|\tilde{\sigma}_i| : |\Delta^n| \to f^{-1}(|\mathrm{sk}_n(X)|)$ of $\mathrm{Sing}(f^{-1}(|\mathrm{sk}_n(X)|))$ under the Yoneda embedding. Here $|\tilde{\sigma}_i|$ is unique continuous map such that the composition $|\Delta^n| \xrightarrow{|\tilde{\sigma}_i|} f^{-1}(|\mathrm{sk}_n(X)|) \hookrightarrow S$ equals $|\sigma|'_i$. To see that the latter is well-defined note that the composition

$$|\partial\Delta^n| \xrightarrow{|\phi'_\sigma|\circ|\rho_{i,\sigma}|} |Z^{(n-1)}| \hookrightarrow S$$

equals $|\sigma|_i'|_{|\partial\Delta^n|}$, by construction of $\rho_{i,\sigma}$. To see that $u_{Z^{(n)}}$ exhibits $f^{-1}(|\mathrm{sk}_n(X)|)$ as geometric realization of $Z^{(n)}$ it suffices to prove that

$$\begin{tikzcd}
\coprod_{\substack{\sigma\in X_n^{\mathrm{nd}}\\ i\in F_\sigma}} |\partial\Delta^n| \arrow[r, hook] \arrow[d, "|\phi_\sigma'|\circ|\rho_{i,\sigma}|"] & \coprod_{\substack{\sigma\in X_n^{\mathrm{nd}}\\ i\in F_\sigma}} |\Delta^n| \arrow[d, "(|\tilde\sigma_i|)"'] \\
f^{-1}(|\mathrm{sk}_{n-1}(X)|) \arrow[r, hook] & f^{-1}(|\mathrm{sk}_n(X)|)
\end{tikzcd}$$

is a pushout square. All spaces, which were mentioned in this proof, belong to the category **CGHaus** of compactly generated Hausdorff spaces as this category contains all covering spaces of its objects. In **CGHaus**$/|X|$ there holds a *fibered exponential law*, see [1, Thm. 3.5 (b)] or [15, Prop. 2.4]. Therefore, the pullback along f as an endofunctor of compactly generated Hausdorff spaces over $|X|$ preserves all colimits. The diagram in consideration is obtained by pulling back a pushout square square along f. Namely the pushout square for $|\mathrm{sk}_n(X)|$ that is provided by the skeletral filtration of the CW-complex $|X|$. Since colimits in the category **CGHaus**$/|X|$ are computed in **CGHaus** by [14, Prop. 3.6], our diagram is a pushout diagram, at least in the category **CGHaus**. But the inclusion **CGHaus** $\to$ **Top** admits a right adjoint, given by k-ification, and therefore preservers all colimits.
It remains to be checked that $|p^{(n)}| = f|_{f^{-1}(|\mathrm{sk}_n(X)|)}$. These maps agree on $f^{-1}(|\mathrm{sk}_{n-1}(X)|)$ by the induction hypothesis. Let $\sigma \in X_n^{\mathrm{nd}}, i \in F_\sigma$ and let $\iota_X : |\mathrm{sk}_{n-1}(X)| \hookrightarrow |X|, \iota_S : f^{-1}(|\mathrm{sk}_n(X)|) \hookrightarrow S$ be the inclusions. We constructed the morphism u_{Z^n} precisely such that $\left|\tilde\Phi_{i,\sigma}\right| = |\tilde\sigma_i|$, so the last diagram and the equation

$$f \circ \iota_S \circ |\tilde\sigma_i| = f \circ |\sigma|_i' = |\sigma| = \iota_X \circ |\Phi_\sigma| = \iota_X \circ |p^{(n)} \circ \tilde\Phi_{i,\sigma}| = \iota_X \circ |p^{(n)}| \circ |\tilde\sigma_i|$$

imply $f \circ \iota_s = \iota_X \circ p^{(n)}$, which concludes the induction.
Define $Z := \mathrm{colim}_{n\geq 0} Z^{(n)}$, let $p : Z \to X$ be the colimit of collection of maps $Z^{(n)} \xrightarrow{p^{(n)}} \mathrm{sk}_n(X) \hookrightarrow X$ and let $u_Z : Z \to \mathrm{Sing}(S)$ be the colimit of the maps

$$Z^{(n)} \xrightarrow{u_{Z^{(n)}}} \mathrm{Sing}(f^{-1}(|\mathrm{sk}_n(X)|) \to \mathrm{Sing}(S).$$

To see that u_Z exhibits S as a geometric realization of Z it suffices to check that the canonical map $\mathrm{colim}_{n\geq 0} f^{-1}(|\mathrm{sk}_n(X)|) \to S$ is a homeomorphism. Again by the *fibered exponential law*, in the category **CGHaus** the pullback of the colimit of the diagram

$$\emptyset \hookrightarrow |\mathrm{sk}_0(X)| \hookrightarrow \cdots \hookrightarrow |\mathrm{sk}_n(X)| \hookrightarrow \ldots \tag{6}$$

along f provides a colimit of the diagram (6) with every term pulled back along f separately. Pulling back every term in the diagram 6 separately in **Top** yields

$$\emptyset \hookrightarrow f^{-1}(|\mathrm{sk}_0(X)|) \hookrightarrow \cdots \hookrightarrow f^{-1}(|\mathrm{sk}_n(X)|) \hookrightarrow \cdots \tag{7}$$

Since **CGHaus** $\to$ **Top** reflects limits and every $f^{-1}(|\mathrm{sk}_n(X)|)$ has been shown to be a compactly generated Hausdorff space, as it is a geometric realization of a simplicial set, pulling back every term in the diagram (6) separately in **CGHaus** also yields diagram (7). Because S is a compactly generated Hausdorff space and $\mathrm{colim}_{n\geq 0}|\mathrm{sk}_n(X)| = |X|$, we have that in both categories, **Top** and **CGHaus**, the pullback of the colimit of the diagram (6) along f equals S. Putting all this together, S is the colimit of the diagram (7) in the category **CGHaus** and therefore also in **Top**. So, u_Z exhibits S as a geometric realization of Z. Lastly, $|p|$ and f agree because every point of $S = |Z|$ is contained in the geometric realization of $|Z^{(n)}| = f^{-1}(|\mathrm{sk}_n(X)|)$ for some $n \geq 0$. $\square$

Lemma 3.10. *For any diagram $X \xrightarrow{q} B \xleftarrow{p} Y$ in **sSet** with p a simplicial covering space projection and any continuous map $f : |X| \to |Y|$ with $|p| \circ f = |q|$, there exists a unique lift $g : X \to Y$ of q along p with $|g| = f$.*

Proof. We first do the proof for simplices:

Claim 3.10.1. *For any $n \geq 0$ and $\sigma : \Delta^n \to X$ there exists a unique $\tilde{\sigma} : \Delta^n \to Y$ such that $p \circ \tilde{\sigma} = q \circ \sigma$ and $f \circ |\sigma| = |\tilde{\sigma}|$.*

Let p' and σ' be the canonical projections maps from the pullback $\Delta^n \times_B Y$ of the diagram $\Delta^n \xrightarrow{q\circ\sigma} B \xleftarrow{p} Y$ to Δ^n and Y, respectively. By proposition 3.1 a), p' is an isomorphism restricted to any connected component and by proposition 2.44 this also holds for $|p'|$. Because Δ^n is finite, $|\Delta^n \times_B Y|$ is the pullback of the diagram $|\Delta^n| \xrightarrow{|q|\circ|\sigma|} |X| \xleftarrow{|p|} |Y|$ by remark 2.25. Hence, there exists a unique continuous map $\lambda : |\Delta^n| \to |\Delta^n \times_B Y|$ such that $|\sigma'| \circ \lambda = f \circ |\sigma|$ and $|p'| \circ \lambda = \mathrm{id}_{|\Delta^n|}$. We conclude from the latter equality that there exists a unique $s : \Delta^n \to X \times_B Y$ such that $p' \circ s = \mathrm{id}_{\Delta^n}$ and $|s| = \lambda$. Now $\tilde{\sigma} := \sigma' \circ s$ satisfies

$$p \circ \tilde{\sigma} = \circ\sigma' \circ s = q \circ \sigma \circ p' \circ \sigma' = q \circ \sigma$$

and $|\tilde{\sigma}| = |\sigma' \circ s| = |\sigma'| \circ |s| = |\sigma'| \circ \lambda = f \circ |\sigma|$. In fact, $\tilde{\sigma}$ is the unique morphism with $|\tilde{\sigma}| = f \circ |\sigma|$ because the geometric realization functor is faithful by proposition 2.26. This proves the claim.
For all $n \geq 0$, define $g_n : X_n \to Y_n$ by mapping $\sigma \in X_n$, to the unique $\tilde{\sigma}$ constructed in the above claim. Given any $\alpha : \Delta^m \to \Delta^n$ for any $m, n \geq 0$ and $\sigma : \Delta^n \to X$. Then $\tilde{\sigma} \circ \alpha$ satisfies the conditions in claim 3.10.1 applied to $\sigma \circ \alpha : \Delta^m \to X$. By definition, g_m maps the simplex $\sigma \circ \alpha$ to $\tilde{\sigma} \circ \alpha$, i.e. $g_m(X(\alpha)(\sigma)) = X(\alpha)(g_m(\sigma))$ by the Yoneda lemma. So, the components $(g_n)_{n\geq 0}$ assemble to a morphism $g : X \to Y$.
g satisfies $p \circ g = q$, because this holds on every simplex. Further, for every simplex $\Delta^n \to X$ of X we have $|g| \circ |\sigma| = |\tilde{\sigma}| = f \circ |\sigma|$ and therefore $f = |g|$ by proposition 2.23. Finally, g is the unique map with $|g| = f$, because geometric realization is a faithful functor by proposition 2.26. $\square$

Definition/Corollary 3.11. *Let X be a simplicial set and let us define **Cov**(X) to be the full subcategory of **sSet**$/X$ consisting of covering projections. Then the geometric realization functor induces an equivalence of categories **Cov**$(X) \to$ **Cov**$(|X|)$.*

Proof. First of all, the geometric realization of a simplicial covering projection is a covering space projection by 3.8. The functor $\mathbf{Cov}(X) \to \mathbf{Cov}(|X|), p \mapsto |p|$ is full and faithful by lemma 3.10 and essentially surjective by 3.9. □

Corollary 3.12. *Let $f : (Y, y_0) \to (B, b_0)$ and $p : (Y, x_0) \to (B, b_0)$ be morphism in the category $\boldsymbol{sSet}^*$ with Y connected and p a covering projection. Then there exists a unique lift of f along p if and only if $\pi_1(f)(\pi_1(X, x_0)) \subseteq \pi_1(p)(\pi_1(Y, y_0))$.*

Proof. This follows from lemma 3.10 and the corresponding statement for topological spaces in proposition 3.3. □

3.2. Normal coverings, chain complexes and classifying spaces

For any group G we denote the integral group ring on G by $\mathbb{Z}[G]$.

Construction 3.13. Given a simplicial set X and a subgroup G of $\mathrm{Aut}(X)$. Then both chain complexes $C^{\Delta}(X)$ and $C^{\mathbf{CW}}(X)$ defined in definition 2.46 become chain complexes of left $\mathbb{Z}[G]$-modules via defining $g \cdot \sigma = g(\sigma)$ for all simplices σ of X and extending linearly.

Definition 3.14. Given a morphism of simplicial sets $p : Z \to X$.
Define $\mathrm{Aut}(p) := \mathrm{Aut}_{\mathbf{sSet}/X}(p)$ to be the subgroup of $\mathrm{Aut}(Z)$ consisting of isomorphisms of Z that lift p along p. For any $n \geq 0$ and $\sigma \in X_n$, the group $\mathrm{Aut}(p)$ acts on the connected components of the pullback $\Delta^n \times_X Z$ of $\Delta^n \xrightarrow{\sigma} X \xleftarrow{p} Z$, where $g \in \mathrm{Aut}(p)$ acts by $\pi_0(\mathrm{id}_{\Delta^n} \times_X g)$.
When p is a covering projection, then the elements of $\mathrm{Aut}(p)$ are called *deck transformations.* In that case, we can identify the set $\{\sigma\} \times_{X_n} Z_n$, consisting of $\tilde{\sigma} \in Z_n$ such that $p(\tilde{\sigma}) = \sigma$, with the set of connected components of $\Delta^n \times_X Z$. Explicitly, we identify $\tilde{\sigma}$ with the connected component of $\Delta^n \times_X Z$ that contains the image of the map induced by id_{Δ^n} and $\tilde{\sigma}$. Under this identification $g \in \mathrm{Aut}(p)$ acts on $\{\sigma\} \times_{X_n} Z_n$ by $g \cdot \tilde{\sigma} = g(\tilde{\sigma})$.
We call a simplicial covering projection p *normal*, if the action defined above is transitive for all $n \geq 0$ and $\sigma \in X_n$.

Lemma 3.15. *For any covering projection $p : Z \to X$ of simplicial sets with Z connected, the group Aut(p) acts freely on the connected components of the pullback of the diagram $\Delta^n \xrightarrow{\sigma} X \xleftarrow{p} Z$, for any $n \geq 0$ and $\sigma \in X_n$.*

Proof. This follows from the uniqueness statement in corollary 3.12. □

Lemma 3.16. *A simplicial covering projection $p : Z \to X$ is normal if and only if $|p| : |Z| \to |X|$ is a normal covering space projection. If Z is connected and $z_0 \in Z$, then a simplicial covering projection p is normal if and only if im$(\pi_1(p, z_0))$ is a normal subgroup $\pi_1(X, p(z_0))$.*

Proof. Note that for a normal covering space projections $f : S \to T$ with S and T path-connected, the deck-transformation group acts transitively on the connected components of the preimage of every evenly covered and connected subset of T. Because π_0 commutes with geometric realization, as we described in proposition 2.44, when $|p|$ is normal, then so is p. The converse direction follows because every $x \in |Z|$ is contained in the image of $|\tilde{\sigma}|$ for some simplex $\tilde{\sigma}$ of Z. The last statement already holds on the level of spaces by [8, prop. 1.39]. □

Proposition 3.17. *Given a morphism of simplicial sets $p : Z \to X$ with Z connected. Let $n \geq 0$ and $G := Aut(p)$. For any $\sigma \in X_n$ choose a* ***fixed*** *$\tilde{\sigma} \in Z_n$ with $p(\tilde{\sigma}) = \sigma$, whenever such a lift exists. Define $B_n := \{\tilde{\sigma}\}_{\sigma \in X_n}, B_n^{nd} := \{\tilde{\sigma}\}_{\sigma \in X_n^{nd}}$ to be the collection of the $\tilde{\sigma}$'s for σ iterating through X_n and X_n^{nd}, respectively.*

When p is a normal covering projection, then the $\mathbb{Z}[G]$-modules $C_n^{\Delta}(Z)$ and $C_n^{\mathbf{CW}}(Z)$ are free with basis B_n and B_n^{nd}, respectively.

Proof. When p is a normal covering projection, then $\mathrm{Aut}(p)$ acts freely and transitively on the set $\{\sigma\} \times_{X_n} Z_n$ of $\tau \in Z_n$ with $p(\tau) = \sigma$, where $g \in \mathrm{Aut}(p)$ acts via $g \cdot \tau = g(\tau)$. So, the statement follows from construction 3.13, because $C^{\Delta}(Z)$ and $C^{\mathbf{CW}}(Z)$ are free abelian groups with basis Z_n and Z_n^{nd}, respectively. □

Definition 3.18. Given a group G. We define a simplicial set EG via $EG_n = \prod_{i=0}^{n} G$ with $d_i((g_0, \dots, g_n)) = (g_0, \dots, \hat{g}_i, \dots, g_n)$ and $s_i(g_0, \dots, g_n) = (g_0, \dots, g_i, g_i, \dots, g_n)$ for all $0 \leq i \leq n$. Further, we define BG to be the simplicial set given by the nerve of the category defined by the group G, i.e. $(BG)_0$ only contains a tuple of length 0, denoted $(\,)$, and $(BG)_n = \prod_{i=1}^{n} G$ for $n \geq 1$. Further, for $(g_1, \dots, g_n) \in (BG)_n$ by definition $d_i(g_1, \dots, g_n) = (g_1, \dots, g_i g_{i+1}, g_{i+2}, \dots, g_n)$ for $0 < i < n$, $d_0(g_1, \dots, g_n) = (g_2, \dots, g_n)$, $d_n(g_1, \dots, g_n) = (g_1, \dots, g_{n-1})$ and s_i inserts 1 at the i^{th} spot of $(g_1, \dots, g_n)$. It is well known that both assignments $G \mapsto EG$ and $G \mapsto BG$ define functors $\mathbf{Grp} \to \mathbf{sSet}$.

Proposition 3.19. *Given a group G. Then*

$$q : EG \to BG, \quad (g_0, \dots, g_n) \mapsto (g_0^{-1} g_1, g_1^{-1} \cdot g_2, g_2^{-1} g_3, \dots, g_{n-1}^{-1} \cdot g_n)$$

defines a normal covering space projection of simplicial sets with $Aut(q) = G$ and $|EG|$ contractible.

Proof. G acts on EG from the left via

$$\underline{G} \times EG \to EG, \quad g \cdot (g_0, \dots, g_n) = (g \cdot g_0, g \cdot g_1, \dots, g \cdot g_n).$$

q is given by the composition $EG \to EG/G \xrightarrow{h} BG$, where the first map is the projection onto the orbit space and h is on representatives given by $(g_0, g_1, \dots, g_n) \mapsto (g_0^{-1} g_1, g_1^{-1} g_2, \dots, g_{n-1}^{-1} g_n)$. h is an isomorphism of simplicial sets with inverse given by mapping $(g_1, \dots, g_n)$ to the equivalence class of

$$(1, g_1, g_1 \cdot g_2, g_1 \cdot g_2 \cdot g_3, \dots, g_1 \cdot g_2 \cdot \ldots g_n).$$

$|EG|$ is contractible and $|EG| \to |EG/G|$ a universal and therefore normal topological covering space projection by [8, Exmp. 1B.7]. So, the same holds for q by theorem 3.8 and lemma 3.16. □

Corollary 3.20. *Let $(X, x_0) \in \boldsymbol{sSet}^*$ be connected, G any group and $f : X \to BG$ a morphism such that $\pi_1(f, x_0)$ is surjective. Construct $Z := EG \times_{BG} X$ with canonical projections $p : Z \to X$ and $t : Z \to EG$.*
Then Z is connected and p a normal covering projection. Further, the map $Aut(q) \to Aut(p), g \mapsto g \times_{EG} id_X$ is a group isomorphism and $im\ (\pi_1(p, z_0)) = \ker \pi_1(f, x_0)$ for any $z_0 \in Z$ with $p(z_0) = x_0$.

Proof. This follows from lemma 3.5 together with corollary 3.11 and lemma 3.16 because π_0 and π_1 both factor through geometric realization. $\square$

Now the main idea for the first section of the next chapter is that any normal covering projection $p : Z \to X$ with Z connected should arise by pulling back $q : E(\mathrm{Aut}(p)) \to B(\mathrm{Aut}(p))$ along a suitable chosen $X \to B(\mathrm{Aut}(p))$. We will, in fact, try to find an algorithm that constructs the required map $X \to B(\mathrm{Aut}(p))$ from the knowledge of the normal subgroup $\mathrm{im}(\pi_1(p))$ of $\pi_1(X, x_0)$.

4. Computation

4.1. Covering spaces

In this whole subsection 4.1 we let Z and X be connected simplicial sets and $p : Z \to X$ a normal covering projection, where we denote $G := \operatorname{Aut}(p)$.
For the next construction, recall the notation and definition of vertices of simplices of simplicial sets in definition 2.8. In the following, for $\sigma = (g_0, \dots, g_n) \in (EG)_n$, we will identify the i^{th} vertex $\sigma(i)$ of σ with the group element $g_i \in G$.

Construction/Proposition 4.1. *Let $l : X_0 \to Z_0$ be a set function with $p_0 \circ l = id_{X_0}$. Define $f : Z \to EG$ by sending $\sigma \in Z_n$ to $(g_0, \dots, g_n) \in (EG)_n$, where $g_i \in G$ is the unique deck-transformation such that $g_i^{-1}(\sigma(i)) = l(p(\sigma)(i))$. Then f is a morphism of simplicial sets and the composition $Z \xrightarrow{f} EG \xrightarrow{q} BG$ factors uniquely as $\overline{f} \circ p$ for $\overline{f} : X \to BG$. Further, for any $x_0 \in X_0$ and $z_0 := l(x_0)$, the sequence*

$$0 \to \pi_1(Z, z_0) \xrightarrow{\pi_1(p)} \pi_1(X, x_0) \xrightarrow{\pi_1(\overline{f})} \pi_1(BG) \to 0$$

is exact. Moreover, $X \xleftarrow{p} Z \xrightarrow{f} EG$ satisfies the universal property of the pullback of $X \xrightarrow{\overline{f}} BG \xleftarrow{q} EG$.

Proof. The g_i's in the definition of f exists uniquely because $\operatorname{Aut}(p)$ acts freely and transitively on the fiber of $p(\sigma)(i)$. Further, by the simplicial identity (1)

$$l(p(d_j\sigma(i))) = \begin{cases} l(p(\sigma)(i)) = g_i^{-1}(\sigma(i)) = g_i^{-1}(d_j\sigma(i)), & \text{for } 0 \le i < j \le n, \\ l(p(\sigma)(i+1)) = g_{i+1}^{-1}(\sigma(i+1)) = g_{i+1}^{-1}(d_j\sigma(i)) & \text{for } 0 \le j \le i \le n-1 \end{cases}$$

and therefore $f(d_j(\sigma)) = d_j(f(\sigma))$ for $0 \le j \le n$.
Similarly, by the simplicial identity (2)

$$l(p(s_j\sigma(i))) = \begin{cases} l(p(\sigma)(i)) = g_i^{-1}(\sigma(i)) = g_i^{-1}(s_j\sigma(i)), & \text{for } 0 \le i \le j \le n, \\ l(p(\sigma)(i-1)) = g_{i-1}^{-1}(\sigma(i-1)) = g_{i-1}^{-1}(s_j\sigma(i)) & \text{for } 0 \le j < i \le n+1 \end{cases}$$

and therefore $f(s_j(\sigma)) = s_j(f(\sigma))$ for $0 \le j \le n$. Hence, f defines a morphism of simplicial sets. Moreover, $g_i^{-1}(\sigma(i)) = l(p(\sigma)(i))$ implies $(gg_i)^{-1}(g\sigma(i)) = l(p(\sigma)(i)) = l(p(g\sigma)(i))$ and therefore $f(g\sigma) = g \cdot f(\sigma)$ for all g, σ. Hence $q \circ f$ is constant on the fibers of p and therefore factors as $\overline{f} \circ p$ for a unique $\overline{f} : X \to BG$.
Next, we take $x_0 \in X_0$ and set $z_0 := l(x_0)$ to prove exactness of the given sequence of groups. Since $|p|$ is a covering map, $\pi_1(p, x_0)$ is injective. We have

$$\pi_1(\overline{f}, x_0) \circ \pi_1(p, z_0) = \pi_1(q, (1)) \circ \pi_1(f, z_0) = 0,$$

because $|EG|$ is contractible. Therefore $\operatorname{im}\,(\pi_1(p, z_0)) \subseteq \ker \pi_1(\overline{f}, x_0)$.
For any $g \in G$ we can find a path γ from $|z_0|$ to $|g(z_0)|$ in $|Z|$ using that Z is connected. The path $|q| \circ |f| \circ \gamma$ represents $g \in \pi_1(BG)$, where we identify $\pi_1(BG)$ with G as in [8,

Prop. 1.40]. Now $|q| \circ |f| \circ \gamma = |\overline{f}| \circ |p| \circ \gamma$ proves surjectivity of $\pi_1(\overline{f})$.
Next, take any loop γ in $|X|$ based at $|x_0|$ such that $|\overline{f}| \circ \gamma$ is null-homotopic. γ lifts to a unique path $\tilde{\gamma}$ in $|Z|$ starting at $|z_0|$. $\tilde{\gamma}$ has endpoint $|g(z_0)|$ for a unique $g \in \mathrm{Aut}(p)$. Since $|f| \circ \tilde{\gamma}$ lifts the null-homotopic loop $|\overline{f}| \circ \gamma$ along the covering map $|q|$, $|f| \circ \tilde{\gamma}$ is a loop based at $|(1)|$, using the homotopy lifting property from proposition 3.3 b). Since $(1) = |f| \circ \tilde{\gamma}(1) = f(g(z_0)) = (g)$, we conclude $g = 1$, so $[\tilde{\gamma}] \in \pi_1(Z, l(x_0))$. Because $\pi_1(p)([\tilde{\gamma}]) = [\gamma]$,we conclude $\ker \pi_1(\overline{f}) \subseteq \mathrm{im}\ \pi_1(p)$, which concludes the proof of exactness of the sequence in consideration.
For the last statement, let A be a simplicial set and $\alpha : A \to X$, $\beta : A \to EG$ be morphisms such that $\overline{f} \circ \alpha = q \circ \beta$. We prove that there exists a unique morphism $t : A \to Z$ with $p \circ t = \alpha$ and $f \circ t = \beta$. By passing to connected components (every simplicial set is the coproduct of its connected components by proposition 2.37), we may assume that A is connected. Because $\pi_1(\overline{f}) \circ \pi_1(\alpha) = \pi_1(q) \circ \pi_1(\beta) = 0$, since $|EG|$ is contractible, we have $\mathrm{im}(\pi_1(\alpha)) \subseteq \ker \pi_1(\overline{f}) = \mathrm{im}\ \pi_1(p)$.
Let a_0 be 0-simplex of A. There exists a unique $g \in G$ with $\beta(a_0) = (g)$. We conclude from corollary 3.12, that there exists a morphism $t : (A, a_0) \to (Z, g \cdot l(\alpha(a_0)))$ such that $p \circ t = \alpha$. Now $\beta : (A, a_0) \to (EG, \beta(a_0))$ and $f \circ t : (A, a_0) \to (EG, (g))$ both lift $\overline{f} \circ \alpha$ along q and therefore agree by the uniqueness assertion in corollary 3.12. Any other morphism $\overline{t}$ with $p \circ \overline{t} = \alpha$ and $f \circ \overline{t} = \beta$ must send a_0 to a lift of $\alpha(a_0)$. Because p is normal, there exists $g' \in G$ with $\overline{t}(a_0) = g' \cdot l(\alpha(a_0))$. Now

$$(g') = f(g' \cdot l(\alpha(a_0))) = f\overline{t_0}(a_0) = \beta(a_0) = (g)$$

implies $g' = g$. So, both $t, \overline{t} : (A, a_0) \to (Z, g \cdot l(\alpha(a_0)))$ lift α along p and therefore agree by the uniqueness assertion in corollary 3.12. All in all, $X \xleftarrow{p} Z \xrightarrow{f} EG$ satisfies the universal property of the pullback of $X \xrightarrow{\overline{f}} BG \xleftarrow{q} EG$. □

Lemma 4.2. *Let $l : X_0 \to Z_0$ be a set function with $p_0 \circ l = id$ and construct $\overline{f}$ and f as in construction 4.1. Let $n \geq 2$, $\sigma \in X_n$.*
Define $\tau := d_0(\sigma)$ and $\rho := d_n(\sigma)$. Let $\tilde{\sigma} \in Z_n$, as well as, $\tilde{\tau}, \tilde{\rho} \in Z_{n-1}$ such that $\tilde{\sigma}(0) = l(\sigma(0))$, $\tilde{\tau}(0) = l(\tau(0))$, $\tilde{\rho}(0) = l(\rho(0))$ and $p(\tilde{\sigma}) = \sigma, p(\tilde{\tau}) = \tau$, $p(\tilde{\rho}) = \rho$.
Then for $i = 0, \dots, n-1$

$$f(\tilde{\sigma})(i) = f(\tilde{\rho})(i) \text{ and } f(\tilde{\sigma})(n) = f(\tilde{\rho})(n-1) \cdot (f(\tilde{\tau})(n-2))^{-1} \cdot f(\tilde{\tau})(n-1). \quad (8)$$

Moreover, $\overline{f}(\sigma) \in G^n$ can be obtained from $\overline{f}(\rho) \in G^{n-1}$ by adding the $(n-1)^{th}$ entry of $\overline{f}(\tau)$ at the n^{th} spot.

Proof. Both $d_n(\tilde{\sigma})$ and $\tilde{\rho}$ lift $d_n(\sigma)$ along ρ and have 0th-vertex $l(\sigma(0))$. Hence, they agree by proposition 3.1 b). For $i = 0, \dots, n-1$

$$\tilde{\sigma}(i) = d_n(\tilde{\sigma})(i) = \tilde{\rho}(i) = [f(\tilde{\rho})(i)](l(\rho(i))) = [f(\tilde{\rho})(i)](l(\sigma(i))), \quad (9)$$

using the definition of f in the third equality. Because p is normal and $p(\tilde{\tau}) = p(d_0(\tilde{\sigma}))$, there exists a unique $g \in \mathrm{Aut}(p)$ with $d_0(\tilde{\sigma}) = g(\tilde{\tau})$. For $i = 1, \dots, n$

$$\begin{aligned}\tilde{\sigma}(i) &= d_0(\tilde{\sigma})(i-1) = g(\tilde{\tau})(i-1) = g(\tilde{\tau}(i-1)) = g([f(\tilde{\tau})(i-1)](l(\tau(i-1)))) \\ &= [g \circ f(\tilde{\tau})(i-1)](l(\sigma(i))),\end{aligned} \quad (10)$$

using the definition of f in the fourth equality.

For $i = 0, \dots, n-1$, equation (9) implies by definition of f that $f(\tilde{\sigma})(i) = f(\tilde{\rho})(i)$. Because the action of $\mathrm{Aut}(p)$ is free on the fibers of p, the equation

$$[f(\tilde{\rho})(n-1)]l(\sigma(n-1)) \overset{(9)}{=} \tilde{\sigma}(n-1) \overset{(10)}{=} [g \circ f(\tilde{\tau})(n-2)](l(\sigma(n-1)))$$

implies $g \circ f(\tilde{\tau})(n-2) = f(\tilde{\rho})(n-1)$. For $i = n$, equation (10) implies $f(\tilde{\sigma}(n)) = g \circ f(\tilde{\tau})(n-1)$ by definition of f. The combination of the previous two results yields

$$f(\tilde{\rho})(n-1) \cdot (f(\tilde{\tau})(n-2))^{-1} \cdot f(\tilde{\tau})(n-1) = g \circ f(\tilde{\tau})(n-1) = f(\tilde{\sigma}(n)).$$

This proves equation (8). By proposition 4.1 $\overline{f}(\sigma) = q(f(\tilde{\sigma}))$, $\overline{f}(\tilde{\rho}) = q(f(\tilde{\rho}))$ and $\overline{f}(\tilde{\tau}) = q(f(\tilde{\tau}))$. So the last statement of this lemma follows from the explicit form of q in proposition 3.19 and the given computation of $f(\tilde{\sigma})$. □

Recall the definition of **Graph**(X) in 2.41. From now on, throughout this subsection 4.1, Γ shall denote a subgraph of **Graph**(X), such that, when we consider **Graph**(X) as an undirected graph with subgraph Γ, Γ is a spanning tree in **Graph**(X).

Lemma 4.3. *There exists a set function $l : X_0 \to Z_0$ with $p_0 \circ l = id_{X_0}$ such that for all $e \in Edges(\Gamma)$ and all $\tilde{e} \in Z_1$ the following holds: $p(\tilde{e}) = e$ and $d_0(\tilde{e}) = l(d_0(e))$ implies $d_1(\tilde{e}) = l(d_1(e))$.*

Proof. Start with any $z_0 \in Z_0$ and define $x_0 := p(z_0)$. Now let $x \in X_0$ be arbitrary. Then there exist unique $e_0, \dots, e_k \in \mathrm{Edges}(\Gamma)$ with $d_0(e_0) = x_0$, $d_1(e_k) = x$ and $d_1(e_i) = d_0(e_{i+1})$ for all $i = 0, \dots, k-1$. Invoking proposition 3.1 b), we might define $\tilde{e}_0$ to be the unique lift of e_0 with $d_0(\tilde{e}_0) = z_0$ and proceed, inductively, to let $\tilde{e}_i$ be the unique 1-simplex of Z such that $d_0(\tilde{e}_i) = d_1(\tilde{e}_{i-1})$ and $p(\tilde{e}_i) = e_i$ for $i = 1, \dots, n$. We define $l(x) := d_1(\tilde{e}_n)$. To see that this l satisfies the required property, note that for any edge $e \in \mathrm{Edges}(\Gamma)$ the unique paths in Γ from x_0 to $d_1(e)$ and to $d_0(e)$, respectively, only differ by traversing e at the end. □

From know on, throughout this subsection 4.1, l shall denote a function satisfying the condition in lemma 4.3. Additionally, we fix a $x_0 \in X_0$ and set $z_0 := l(x_0)$. Further, for the remainder of this subsection 4.1, we define $\{e_i\} \subseteq X_1^{\mathrm{nd}}$ to be the set of nondegenerate 1-cells of X, which are not edges of Γ, i.e. $\mathrm{Edges}(\mathrm{Graph}(X)) \setminus \mathrm{Edges}(\Gamma) = \{e_i\}$.
Recall that in proposition 2.49 we have constructed a normal subgroup R of the free group $F(\{e_i\}_{i \in I})$ and a group epimorphism $\alpha : F(\{e_i\}) \to \pi_1(X, x_0)$ with kernel R. For the remainder of this subsection 4.1, choose a collection $\{n_i\} \subseteq F(\{e_i\})$ such that the normal subgroup generated by $\{n_i\} \cup R$ in $F(\{e_i\})$ equals $N := \alpha^{-1}(\pi_1(p)(\pi_1(Z, z_0)))$.
It is easy to imagine what is going on when Γ is contracted to a point. Then the e_i's are given by the nondegenerate edges of X and form loops. The n_i's are elements of $\pi_1(X, x_0)$ represented by a finite concatenation of the loops e_i, possibly traversed backwards. We require that the normal subgroup generated by $\alpha(\{n_i\})$ in $\pi_1(X, x_0)$ equals $\pi_1(p)(\pi_1(Z, z_0))$.
Because p is a normal covering projection, there is homomorphism $L_{p,z_0} : \pi_1(X, x_0) \to$

$G = \mathrm{Aut}(p)$, which sends the equivalence class of $\gamma : (I, \partial I) \to (|X|, |x_0|)$ to the unique $g \in \mathrm{Aut}(p)$ such that the unique $\tilde{\gamma} : (I, 0) \to (|Z|, |z_0|)$ lifting γ has endpoint $|g(z_0)|$. By proposition 3.3 c) and corollary 3.11 L_{p,z_0} is surjective and has kernel $\pi_1(p)(\pi_1(Z, z_0))$.

Lemma 4.4. *Let $e_i \in \mathit{Edges}(\mathit{Graph}(X)) \setminus \mathit{Edges}(\Gamma)$ and $\tilde{e} \in Z_1^{nd}$ with $p(\tilde{e}) = e_i$ and $\tilde{e}(0) = l(\tilde{e}(0))$. Then*

$$\tilde{e}(1) = (L_{p,z_0} \circ \alpha(e_i))(l(e_i(1))) \text{ and therefore } f(\tilde{e}) = (1, L_{p,z_0} \circ \alpha(e_i)).$$

Proof. Choose a loop $\gamma : (I, \partial I) \to (|X|, |x_0|)$ such that $\gamma(I) \subseteq |\Gamma| \cup |e_i|$ and such that there are unique $s, t \in I$ with $\gamma(t) = |e_i(0)|$ and $\gamma(s) = |e_i(1)|$. Further arrange that $t < s$ and $\gamma([t, s]) = |e_i|$. Then $\alpha(e_i) = [\gamma]$ by the explicit construction of α in proposition 2.49. Let $\tilde{\gamma} : (I, 0) \to (|Z|, |z_0|)$ be a lift of γ along the covering map $|p|$.
Then $g := L_{p,z_0} \circ \alpha(e_i)$ is the unique deck-transformation of p sending z_0 to $\tilde{\gamma}(1)$. Because $\tilde{\gamma}|_{[0,t]}$ is a path only living on edges lifting edges of Γ and $\tilde{\gamma}(0) = |l(x_0)|$, lemma 4.3 yields $\tilde{\gamma}(t) = |l(e_i(0))| = |\tilde{e}(0)|$. We obtain $\tilde{\gamma}([t, s]) = \mathrm{im}(|\tilde{e}|)$ because $\tilde{e}$ is the unique lift of e with $\tilde{e}(0) = l(e_i(0))$. Therefore, $\tilde{\gamma}(s) = |\tilde{e}(1)|$.
Because $|g|^{-1}(\tilde{\gamma}(1)) = |l(x_0)|$ and $|g|^{-1} \circ \tilde{\gamma}|_{[s,1]}$ lives on edges lifting edges of Γ, lemma 4.3 yields $|g|^{-1} \circ \tilde{\gamma}(s) = |l(e_i(1))|$. All in all, $|\tilde{e}(1)| = \tilde{\gamma}(s) = |g(l(e_i(1)))|$ and hence $\tilde{e}(1) = g(l(e_i(1)))$, which is what we wanted to prove. □

Corollary 4.5. *The map $\overline{f} : X \to BG$ satisfies: $\overline{f}(v) = ()$ for all $v \in X_0$ and*

$$\overline{f}(e) = \begin{cases} (1), & e \in \mathit{Edges}(\Gamma), \\ ((L_{p,z_0} \circ \alpha)(e)), & \mathit{else}, \end{cases}$$

for all $e \in X_1^{nd}$. For all $n \geq 2$ and $\sigma \in X_n$, the tuple $\overline{f}(\sigma)$ is obtained from the $(n-1)$-tuple $\overline{f}(d_n(\sigma))$ by adding the $(n-1)^{th}$ entry of $\overline{f}(d_0(\sigma))$ at the n^{th} spot.

Proof. BG has only one vertex, so the first statement is clear. Lemma 4.3 guaranties that all $e \in \mathrm{Edges}(\Gamma)$ are send to (1). Lemma 4.4 treats the case of nondegenerate 1-simplices not contained in $\mathrm{Edges}(\Gamma)$. For higher dimensional simplices this is implied by the last statement in lemma 4.2. □

$L_{p,z_0} \circ \alpha : F(\{e_i\}) \to G$ is surjective and has kernel N. Hence there is a unique isomorphism $\phi : G' := F(\{e_i\}_{i \in I})/N \to G$ such that the composition $F(\{e_i\}) \xrightarrow{\beta} G' \xrightarrow{\phi} G$ equals $L_{p,z_0} \circ \alpha$. Here β denotes the canonical projection. By proposition 4.1, p is the pullback of $\overline{f}$ along q, therefore pulling back the covering map $q' : EG' \to BG'$ from proposition 3.19 along $(B\phi)^{-1} \circ \overline{f}$ yields a morphism $p' : Z' \to X$, which is fiber-wise isomorphic to p. Choose z_0' with $p'(z_0') = x_0$. Because p is a covering projection, also $p' : Z' \to X$ is a covering projection. Moreover, p' satisfies $\pi_1(p')(\pi_1(Z', z_0')) = \pi_1(p)(\pi_1(Z, z_0)) = N$. Because we are in the following only interested in the isomorphism type of $p : Z \to X$ in the category of simplicial sets over X, we will **assume from now on that**

$$G = G' \text{ and } \phi = \mathrm{id}_G. \text{ So, consequently } L_{p,z_0} \circ \alpha = \beta,\ q' = q,\ Z' = Z \text{ and } p' = p. \quad (11)$$

Now suppose we are given N, but not p. Then we can still do construction 2.49 to obtain $\pi_1(X)$ and define β as the quotient map $\pi_1(X) \to \pi_1(X)/N$. This is implemented in the first part of algorithm 1. Corollary 4.5 immediately gives an algorithm to obtain $\overline{f}$ from β. This is implement in the second part of algorithm 1. The third part of the algorithm then constructs $p : Z \to X$ by pulling back $q : EG \to BG$ along $\overline{f}$.

Remark 4.6 (Implementing simplicial sets). Note that the given algorithm only "sees" nondegenerate simplices, since this is the standard implementation of simplicial sets, as in [17]. The used data structure implementing a finite simplicial set X consists of the set of nondegenerate simplices of X, where for each nondegenerate simplex σ its dimension n and its faces are specified. The i^{th} face of σ is specified by a nondegenerate simplex ρ of X and $I \subseteq [n-1]$ such that $d_i\sigma = s_I\rho$. Here we use the notation of convention 2.12. That this data structure uniquely determines X is proven in proposition 2.11. Such a data structure describes a simplicial set if and only if the simplicial identities (1), (2), (3) are satisfied.

Remark 4.7. When we use algorithm 1 with $N = 0$ then we actually construct the universal cover of X. The reason for requiring in algorithm 1 the index $[\pi_1(X) : N]$ to be finite is that $\mathrm{sk}_{\dim(X)}(EG)$ and $\mathrm{sk}_{\dim(X)}(BG)$ are only then finite simplicial sets. Moreover, I want to remark that the image of $\pi(p)(\pi_1(Z, z_0))$ doesn't change if we alter x_0 because this corresponds to conjugating the normal subgroup N. The same applies if we do the construction of l in lemma 4.3 by starting with a different $z_0 \in Z_0$. This justifies that we omit basepoints in algorithm 1.

Let $n \geq 0$. By proposition 3.1 b) for any $\sigma \in X_n$ there exists a unique $\tilde{\sigma} \in Z_n$ with $p(\tilde{\sigma}) = \sigma$ and $\tilde{\sigma}(0) = l(\sigma(0))$. Because p is normal, the map

$$G \times X_n \to Z_n, \quad (g, \sigma) \mapsto g(\tilde{\sigma}) \tag{12}$$

is a bijection. By proposition 3.1 b) the map also restricts to a bijection $G \times X_n^{\text{nd}} \to Z_n^{\text{nd}}$. The following lemma gives a precise answer to how this map fails to be natural.

Lemma 4.8. *Let $n \geq 1$, $\sigma \in X_n$, $0 \leq i \leq n$ and $\tau := d_i(\sigma)$. By proposition 3.1 b), there is a unique $\tilde{\sigma} \in Z_n$ with $p\tilde{\sigma} = \sigma$ and $\tilde{\sigma}(0) = l(\sigma(0))$, as well as, a unique $\tilde{\tau} \in Z_{n-1}$ with $p\tilde{\tau} = \tau$ and $\tilde{\tau}(0) = l(\tau(0))$. Then for any $g \in G$*

$$d_i(g(\tilde{\sigma})) = (g \circ g_i)(\tilde{\tau}) \text{ with } g_i = \begin{cases} f(\tilde{\sigma})(1), & \text{if } i = 0, \\ 1, & \text{else.} \end{cases} \tag{13}$$

In the case $n \geq 2$, we additionally define $\rho := d_n(\sigma)$ and $\lambda := d_0(\rho)$. By proposition 3.1 b), there is a unique $\tilde{\rho} \in Z_{n-1}$ with $p\tilde{\rho} = \rho$ and $\tilde{\rho}(0) = l(\rho(0))$, as well as, a unique $\tilde{\lambda} \in Z_{n-2}$ with $p\tilde{\lambda} = \lambda$ and $\tilde{\lambda}(0) = l(\lambda(0))$. Then $d_0(\tilde{\rho}) = g_0(\tilde{\lambda})$. Moreover, when $\rho = s_0\rho'$ for some $\rho' \in X_{n-2}$, then $g_0 = 1$.
For $n = 1$, we have

$$g_0 = \begin{cases} \beta(\sigma), & \text{if } \sigma \text{ is nondegenerate and } \sigma \in \mathit{Edges}(\mathit{Graph}(X)) \setminus \mathit{Edges}(\Gamma), \\ 1, & \text{else.} \end{cases}$$

Algorithm 1 Covering space for given normal subgroup of $\pi_1(X)$ with finite index

```
Require: $X$: Finite connected simplicial set,
         $\{e_i\}_{1\le i\le m}$: $X_1^{\mathrm{nd}} \setminus \{e_1,\dots,e_m\}$ is set of edges of a spanning tree $\Gamma$ in $\mathrm{sk}_1(X)$,
         $\{n_i\}$: elements of $F(e_1,\dots,e_m)$ generating $N \lhd \pi_1(X)$ with $[\pi_1(X):N] < \infty$.
         (Here we identify $\pi_1(X)$ with a quotient of $F(e_1,\dots,e_m)$ via prop. 2.49.)
Ensure: $G$: $\pi_1(X)/N$, $p$: Covering projection $Z \to X$ with $\mathrm{im}(\pi_1(p)) = N$,
        $\overline{f}$: Morphism $X \to BG$ with $\pi_1(f)$ surjective and $\ker \pi_1(f) = N$,
        $f$: Morphism $Z \to EG$ with $X \xleftarrow{p} Z \xrightarrow{f} EG$ the pullback of $X \xrightarrow{\overline{f}} BG \leftarrow EG$

Part 1: Construction of a presentation of $G$

  Define $F :=$ FreeGroup$(e_1,\dots,e_m)$.
  rels := [ ]     ▷ This empty list will contain relations s.t. $F(e_1,\dots,e_n)/rels = \pi_1(X)$
  for $\sigma \in X_2^{\mathrm{nd}}$ do
      Define $z := 1 \in F$.     ▷ This is the relation that $\sigma$ adds to $\pi_1(X)$
      for $i = 0,1,2$ do
          if $d_i\sigma = e_j$ for some $j = 1,\dots,m$ then
              Set $z = z * (e_j)^{(-1)^i}$.
      rels.append($z$)
  Define $\beta : F \to G := F/ < \mathrm{rels} \cup \{n_1,\dots,n_s\} >$ to be the canonical projection.

Part 2: Construction of $\overline{f} : X \to BG$

  $\overline{f}(v) = (\,) \in BG_0$ for all $v \in X_0$   ▷ The empty tuple shall be the unique vertex of BG
  for $e \in X_1^{\mathrm{nd}}$ do
      if $e = e_j$ for some $j = 1,\dots,m$ then
          $\overline{f}(e) = (\beta(e_j))$
      else
          $\overline{f}(e) = (1)$
  for $n = 2,\dots,\dim X$ do
      for $\sigma \in X_n^{\mathrm{nd}}$ do
          $s_I(\tau) := d_0(\sigma)$, $s_J(\rho) := d_n(\sigma)$     ▷ $\tau$, $\rho$ nondegenerate as in 2.12
          $\overline{f}(\sigma)$ is the $n$-tuple obtained from the $(n-1)$-tuple $s_J\overline{f}(\rho)$ by adding the last
          entry of $s_I\overline{f}(\tau) \in G^{n-1}$ at the $n^{\mathrm{th}}$ spot.

Part 3: Construction of the covering map

  Construct the covering map $q : \mathrm{sk}_{\dim X}(EG) \to \mathrm{sk}_{\dim X}(BG)$.
  Obtain $X \xleftarrow{p} Z \xrightarrow{f} EG$ as the pullback of $X \xrightarrow{\overline{f}} \mathrm{sk}_{\dim X}(BG) \xleftarrow{q} \mathrm{sk}_{\dim X}(EG)$
  return $G, p, \overline{f}, f$
```

Proof of lemma 4.8. For $i > 0$ taking the 0th-vertex of a simplex commutes with d_i. So, we have $\tau(0) = \sigma(0)$ and therefore $\tilde{\sigma}(0) = \tilde{\tau}(0)$, as well as

$$d_i(g(\tilde{\sigma}))(0) = g(\tilde{\sigma}(0)) = g(\tilde{\tau})(0).$$

In conclusion, $d_i(g(\tilde{\sigma}))$ and $g(\tilde{\tau})$ have the same 0th-vertex. As both of these simplices lift $d_i\sigma = \tau$ along the simplicial covering map p and have the same 0th-vertex, they agree by proposition 3.1 b).
On the other hand, $\tau(0) = d_i(\sigma)(0) = \sigma(1)$. Define $g_0 := f(\tilde{\sigma})(1)$. Then, by definition of f in construction 4.1, we have

$$\tilde{\sigma}(1) = g_0(l(\sigma(1))) = g_0(l(\tau(0))) = g_0(\tilde{\tau}(0)). \tag{14}$$

Both $g \circ g_0(\tilde{\tau})$ and $d_0(g(\tilde{\sigma}))$ lift $\tau = d_0(\sigma)$ along the simplicial covering map p and have the same 0th-vertex by the following equation:

$$d_0(g(\tilde{\sigma}))(0) = g(\tilde{\sigma}(1)) \overset{(14)}{=} (g \circ g_0)(\tilde{\tau})(0)$$

Hence, $d_0(g(\tilde{\sigma})) = (g \circ g_0)(\tilde{\tau})$ by proposition 3.1 b). This concludes the proof of the first statement of this lemma.
Now assume we are in the situation described for the case $n \geq 2$. The 0th vertex of $d_0(\tilde{\rho})$ equals $\tilde{\rho}(1) = \tilde{\sigma}(1)$ and the 0th vertex of λ equals $\rho(1) = \sigma(1)$. We have already seen that $\tilde{\sigma}(1) = g_0(l(\sigma(1)))$. Putting these observations together we obtain

$$d_0(\tilde{\rho})(0) = \tilde{\sigma}(1) = g_0(l(\sigma(1))) = g_0(l(\lambda(0))) = g_0(\tilde{\lambda}(0)) = g_0(\tilde{\lambda})(0).$$

Both $d_0(\tilde{\rho})$ and $g_0(\tilde{\lambda})$ lift $d_0(\rho) = \lambda$ along the simplicial covering map p and have the same 0th-vertex, hence these simplices agree by proposition 3.1 b).
In the case $\rho = s_0\rho'$ for some $\rho' \in X_{n-2}$, then $\sigma(0) = \sigma(1)$ and $\tilde{\sigma}(0) = \widetilde{\sigma(1)}$. Therefore

$$g_0(l(\sigma(1))) = \tilde{\sigma}(1) = \tilde{\sigma}(0) = l(\sigma(0)) = l(\sigma(1)),$$

so $g_0 = 1$ by lemma 3.15 applied to $\sigma(1) \in X_0$.
Now assume $n = 1$. For $\sigma \in \text{Edges}(\Gamma)$ we have $f(\tilde{\sigma}) = (1, 1)$ because l was chosen to satisfy the properties in 4.3. In the case that σ is nondegenerate and σ is not a edge of Γ, we have computed $f(\tilde{\sigma})$ in lemma 4.4 to be $(1, L_{p,z_0} \circ \alpha(\sigma))$. But convention (11) says $L_{p,z_0} \circ \alpha = \beta$. By the already proven part of the lemma, $g_0 = f(\tilde{\sigma})(1)$, which equals $L_{p,z_0} \circ \alpha(\sigma) = \beta(\sigma)$. The only remaining case is that σ is a degenerate 1-simplex and then $\tilde{\sigma}(1) = \tilde{\sigma}(0)$ by proposition 3.1 b). The equation $\tilde{\sigma}(1) = \tilde{\sigma}(0) = l(\sigma(0))$ implies by the definition of f that $f(\tilde{\sigma})(1) = 1$. But we have already seen that $g_0 = f(\tilde{\sigma})(1)$. □

For all $n \geq 0$, the set of nondegenerate n-simplices of Z is in bijection to $X_n^{\text{nd}} \times G$ by (12). By lemma 4.8, the faces of a 1-simplex in Z are determined by, firstly, the faces of its image under p in X and, secondly, β. Again by lemma 4.8, for $n \geq 2$, the faces of any $n-$simplex in Z_n are determined by, firstly, the faces of its image in X under p, secondly, G, and, thirdly, the faces of all simplices in Z of dimension $< n$. In this way we obtain an algorithm that computes Z in the sense of remark 4.6 by induction on n from the knowledge of X, G and β. In algorithm 2 we record the pseudocode for this. Instead of taking β and G as input, in algorithm 2, β and G are constructed from the sets $\{n_i\}$ and $\{e_i\}$ as in the first part of (1).

Now suppose we are interested in the chain-complex $C^{\mathbf{CW}}(Z)$ from construction 3.13. We know for all $n \geq 0$ from proposition 3.17 that the $\mathbb{Z}[G]$-module $C^{\mathbf{CW}}(X)_n$ has basis $\{\tilde{\sigma}\}_{\sigma \in X_n}$, where $p\tilde{\sigma} = \sigma$ and $\tilde{\sigma}(0) = l(\sigma(0))$ for all $\sigma \in X_n$. So, to compute $C^{\mathbf{CW}}(Z)$ it suffices to give the matrices representing the boundary operators with respect to this basis. We can read of the coefficients of $d(\tilde{\sigma})$ with respect to this basis from lemma 4.8, provided that β, G and X is known. Again, we can compute β and G as in the first part of algorithm 1 from the sets $\{e_i\}$ and $\{n_i\}$. We obtain algorithm 3, which computes $C^{\mathbf{CW}}(Z)$, when X and the sets $\{e_i\}$ and $\{n_i\}$ are given.

Remark 4.9. The given algorithms 1, 2 and 3 require the user to compute a spanning tree and to input the sets $\{e_i\}$, $\{n_i\}$ depending on that tree. In some cases it might be convenient for the user to provide the normal subgroup $N \trianglelefteq \pi_1(X)$ by loops in the 1-skeleton of the simplicial set X instead of computing Γ. For this purpose we can modify the first part of these algorithms, by replacing them as in algorithm 4.

Algorithm 2 Covering space for given normal subgroup of $\pi_1(X)$ with finite index

```
Require: $X$: Finite connected simplicial set,
         $\{e_i\}_{1\le i\le m}$: $X_1^{\mathrm{nd}} \setminus \{e_1,\dots,e_m\}$ is set of edges of a spanning tree $\Gamma$ in $\mathrm{sk}_1(X)$,
         $\{n_i\}$: elements of $F(e_1,\dots,e_m)$ generating $N \lhd \pi_1(X)$ with $[\pi_1(X) : N] < \infty$.
         (Here we identify $\pi_1(X)$ with a quotient of $F(e_1,\dots,e_m)$ via prop. 2.49.)
Ensure: $p$: Covering projection $Z \to X$ with $\mathrm{im}\ (\pi_1(p)) = N$
```

Part 1: Construction of a presentation of the group G

```
  This is exactly the same as in algorithm 1.
```

Part 2: Construction of Z

```
  For all $v \in X_0$ and all $g \in G$ define a 0-simplex $g \cdot \tilde{v} \in Z_0$.
  for $e \in X_1^{\mathrm{nd}}$ and $g \in G$ do
      $v := d_0(e), w := d_1(e)$
      if $e = e_j$ for some $j = 1,\dots,m$ then
          Define a 1-simplex $g \cdot \tilde{e}$ with $d_0(g \cdot \tilde{e}) = (g \cdot \beta(e_j)) \cdot \tilde{v}$ and $d_1(g \cdot \tilde{e}) = g \cdot \tilde{w}$.
      else
          Define a 1-simplex $g \cdot \tilde{e}$ with $d_0(g \cdot \tilde{e}) = g \cdot \tilde{v}$ and $d_1(g \cdot \tilde{e}) = g \cdot \tilde{w}$.
  for $n = 2,\dots,\dim(X)$ do
      for $\sigma \in X_n^{\mathrm{nd}}$ and $g \in G$ do
          $d_i(\sigma) = s_{I_i}(\tau_i)$ for $i = 0,\dots,n$, $d_0(\tau_n) = s_J(\lambda)$ ▷ $\tau_i, \lambda$ nondegenerate as in 2.12
          if $0 \notin I_n$ then
              Obtain $g_0 \in G$ from $d_0(1 \cdot \tilde{\tau_n}) = s_J(g_0 \cdot \tilde{\lambda})$, which has already been set.
          else
              Set $g_0 := 1 \in G$.
          Define an $n$-simplex $g \cdot \tilde{\sigma}$ with:
          $d_0(\tilde{\sigma}) = s_{I_0}((g \cdot g_0) \cdot \tau_0)$ and $d_i(g \cdot \tilde{\sigma}) = s_{I_i}(g \cdot \tau_i)$ for $i = 1,\dots,n$.
  Define $Z$ as the simplicial set with $Z_n^{\mathrm{nd}} = \{g \cdot \tilde{\sigma} : g \in G, \sigma \in X_n^{\mathrm{nd}}\}$ for all $n \ge 0$.
  Define $p : Z \to X$ via $g \cdot \tilde{\sigma} \mapsto \sigma$ for all $n \ge 0, g \in G$ and $\sigma \in X_n^{\mathrm{nd}}$.
  return $p$
```

Algorithm 3 Chain complex of covering space for given normal subgroup N of $\pi_1(X)$

Require: X: connected simplicial set, $X_n^{\mathrm{nd}} = \{\sigma_1^{(n)}, \ldots, \sigma_{m_n}^{(n)}\}$ for all $n \leq \dim X < \infty$
$\{e_i\}_{1 \leq i \leq m}$: $X_1^{\mathrm{nd}} \setminus \{e_1, \ldots, e_m\}$ is set of edges of a spanning tree Γ in $\mathrm{sk}_1(X)$,
$\{n_i\}$: set of elements of $F(e_1, \ldots, e_m)$ generating $N \lhd \pi_1(X)$.

Ensure: $\{M_i\}_{1 \leq i \leq \dim(X)}$: Matrices representing the boundary operators of the $\mathbb{Z}[G]$-module chain complex $C^{\mathbf{CW}}(Z)$ with respect to a basis as in proposition 3.17. Here Z is a covering space of X with $\pi_1(Z) \cong N$ and $G := \pi_1(X)/N$.

Part 1: Construction of a presentation of G

This is exactly the same as part 1 in algorithm 1.

Part 2: Construction of $C^{\mathbf{CW}}(Z)$

```
$R := \mathbb{Z}[G]$
deckTrnsf := $\{n : [\,]$ for $n = 1, \ldots, \dim(X)\}$        ▷ encodes the $g_0$'s from algorithm 2
brdyComplx := $\{n : M_n :=$ matrix$(R, m_{n-1}, m_n)$ for $n = 1, \ldots, \dim X\}$        ▷ $M_n := 0$
for $i = 1, \ldots, m_1$ do
    $\sigma_j^{(0)} := d_0\sigma_i^{(1)}$, $\sigma_k^{(0)} := d_1\sigma_i^{(1)}$, $M_1[k,i] -= 1$        ▷ Obtain list indices j, k
    if $\sigma_i^{(1)} = e_r$ for some $r = 1, \ldots, m$ then
        deckTrnsf[1].append($\beta(e_j)$); $M_1[j,i] += \beta(e_j)$
    else
        deckTrnsf[1].append(1); $M_1[j,i] += 1$
for $n = 2, \ldots, \dim X$ do
    for $i = 1, \ldots, m_n$ do
        for $r = 1, \ldots, n$ do
            if $d_r\sigma_i^{(n)} = \sigma_j^{(n-1)}$ is nondegenerate then
                $M_n[j,i] += (-1)^r$
        $d_n\sigma_i^{(n)} = s_I(\sigma_j^{(s)})$ ▷ Obtain degeneracies, dimension s and list index j of $d_n\sigma_i^{(n)}$
        if $0 \in I$ then
            if $d_0(\sigma_i^{(n)}) = \sigma_k^{(n-1)}$ is nondegenerate then
                $M_n[k,i] += 1$
            deckTrnsf[$n$].append(1)
        else
            $g_0 :=$ deckTrnsf[$s$][$j$]; deckTrnsf[$n$].append($g_0$)
            if $d_0(\sigma_i^{(n)}) = \sigma_k^{(n-1)}$ is nondegenerate then
                $M_n[k,i] += g_0$
return brdyComplx
```

Algorithm 4 Modification of the first parts of the algorithms 1, 2 and 3

Require: X: finite connected simplicial set,
$\{n_i\}_{0\leq i<s}$: each n_i a list of tuples $(\sigma_{i_1}, k_{i_1}), \dots, (\sigma_{i_{r_i}}, k_{i_{r_i}})$, where $\sigma_{i_j} \in X_1^{\mathrm{nd}}$ and $k_{i_j} \in \mathbb{Z}$ such that $\sigma_{i_1}^{k_{i_1}} * \cdots * \sigma_{i_{r_i}}^{k_{i_{r_i}}}$ is forming a loop in $\mathrm{sk}_1(X)$.

Ensure: ...

Part 1: Construction of a presentation of G

```
Compute a spanning tree $\Gamma$ in $\mathbf{Graph}(X)$.
$F :=$ FreeGroup$(e_1, \dots, e_m)$ for $X_1^{\mathrm{nd}} \setminus \mathrm{Edges}(\Gamma) = \{e_1, \dots, e_m\}$.
rels := [ ]        ▷ This empty list will contain relations s.t. $F(e_1, \dots, e_n)/rels = \pi_1(X)$
for $\sigma \in X_2^{\mathrm{nd}}$ do
    Define $z := 1 \in F$        ▷ This is the relation that $\sigma$ adds to $\pi_1(X)$
    for $i = 0, 1, 2$ do
        if $d_i\sigma = e_j$ for some $j = 1, \dots, m$ then
            Set $z = z * (e_j)^{(-1)^i}$
    rels.append($z$)
for $i = 0, \dots, s-1$: do
    $N_i := F$.one()
    for $j = 1, \dots, r_i$: do
        if $\sigma_{i_j} = e_l$ for some $l = 1, \dots, m$ then
            Set $N_i = N_i * e_l^{k_{i_j}}$.
Define $\beta : F \to G := F/ <$ rels $\cup \{N_1, \dots, N_s\} >$ to be the projection.
```

...

4.2. Combinatorial torsion of chain complexes

Definition 4.10 (Combinatorial torsion of a chain complex)**.** Let $\mathbb{F}$ be a field or more generally a principal ideal domain, $d \in \mathbb{N}_{>0}$, $m_0, \dots, m_d \in \mathbb{N}_0$ and C a chain complex of $\mathbb{F}$-vector spaces with

$$C_k = \begin{cases} 0, & \text{if } k < 0, \\ \mathbb{F}^{m_k}, & \text{if } 0 \le k \le d, \\ 0, & \text{if } k > d, \end{cases} \quad \text{(we demand \textbf{equality}, not a mere isomorphism)},$$

and boundary maps $\partial_k : C_k \to C_{k-1}$.
We will refer to the three properties that the chain complex C satisfies as *bounded*, *finitely generated* and *based (by the standard basis)*. Finitely generated will be abbreviated by *f.g.* and refers to the chain groups being finite dimensional.
If C is not exact, we define $\tau(C) = 0 \in \mathbb{F}$. When C is **acyclic** we do the following. For all $0 < k \le d+1$ choose a basis $\{b_i^{(k-1)}\}_{0 \le i < r_{k-1}}$ of $\operatorname{im}(\partial_k) \subseteq C_{k-1}$ with $r_{k-1} \in \mathbb{N}_0$. For every $0 < k \le d$ the sequence

$$0 \to \operatorname{im}(\partial_{k+1}) \to C_k \xrightarrow{\partial_k} \operatorname{im}(\partial_k) \to 0$$

is split exact, so there exist $\{b_{i+r_k}^{(k)}\}_{0 \le i < r_{k-1}}$ in $C_k = \mathbb{F}^{m_k}$ such that $\partial_k b_{i+r_k}^{(k)} = b_i^{(k-1)}$ for $0 \le i < r_{k-1}$ and such that $\{b_i^{(k)}\}_{0 \le i < m_k}$ is a basis of $C_k = \mathbb{F}^{m_k}$. We define the *(combinatorial) torsion* of the chain complex C as

$$\tau(C) := \prod_{k=0}^{d} \det(b_0^{(k)}, \dots, b_{m_k-1}^{(k)})^{(-1)^{k+1}} \in \mathbb{F}^*,$$

where $(b_0^{(k)}, \dots, b_{m_k-1}^{(k)}) \in \operatorname{Mat}_{m_k, m_k}$ denotes the $m_k \times m_k$-matrix with i-th column given by the column-vector $b_i^{(k)} \in \mathbb{F}^{m_k}$.

Remark 4.11. The definition of $\tau(C)$ in 4.10 is independent of the choice of the basis $\{b_i^{(k-1)}\}_{0 \le i < r_{k-1}}$ of $\operatorname{im}(\partial_k)$ and the lifts $\{b_{i+r_k}^{(k)}\}_{0 \le i < r_{k-1}}$ in $\mathbb{F}^{m_k}$ for all $0 < k \le d$. A proof of this can be found in [20, lem. 1.3].

Proof of algorithm 5. We have to show for all $0 \le k \le d+1$ that, firstly, the constructed $\{b_i^{(k-1)}\}_{0 \le i < r_{k-1}}$ are a basis of $\operatorname{im}(\partial_k)$ and that, secondly, $\{b_i^{(k-1)}\}_{0 \le i < m_{k-1}}$ forms a basis of $\mathbb{F}^{m_k}$. We prove this by induction on k, where the induction step is backwards from k to $k-1$. Clearly, the statement holds for $k = d+1$. Let us therefore assume that $1 \le k \le d$ and by induction hypothesis we may assume that $\{b_i^{(k)}\}_{0 \le i < r_k}$ is a basis of $\operatorname{im}(\partial_{k+1})$ and that $\{b_i^{(k)}\}_{0 \le i < m_k}$ is a basis of $\mathbb{F}^k$. We know that C_k decomposes as a direct sum of two subspaces, where the first summand given by $\ker(\partial_k) = \operatorname{im}(\partial_{k+1}) = \operatorname{span}(\{b_i^{(k)}\}_{0 \le i < r_k})$ and the second summand is mapped isomorphically to $\operatorname{im}(\partial_k)$ by ∂_k. Since $\{b_i^{(k)}\}_{0 \le i < m_k}$ is a basis of $\mathbb{F}^{m_k}$, the second summand must be given by $\operatorname{span}(b_i^{(k)} : r_k \le i < m_k)$

Algorithm 5 Torsion of a bounded, f.g., based acyclic chain complex over a PID $\mathbb{F}$

Require: C an **acyclic**, bounded, based and finitely generated chain complex over $\mathbb{F}$.
Ensure: $\tau \in \mathbb{F}^*$ the combinatorial torsion of C.

```
1: $r_d := 0$
2: Define $\{b_i^{(d)}\}_{0 \le i < m_d}$ to be the standard basis of $\mathbb{F}^d$.
3: for $k = d, d-1, \dots, 1$ do
4:     $r_{k-1} := m_k - r_k$
5:     $b_i^{(k-1)} := \partial_k b_{i+r_k}^{(k)}$ for $0 \le i < r_{k-1}$.
6:     Extend $\{b_i^{(k-1)}\}_{0 \le i < r_{k-1}}$ to a basis $\{b_i^{(k)}\}_{0 \le i < m_{k-1}}$ of $\mathbb{F}^{m_{k-1}}$.
7: return $\tau := \prod_{k=0}^{d-1} \det(b_0, \dots, b_{m_k-1})^{(-1)^{k+1}}$
```

and therefore the collection $\{\partial_k b_i^{(k)}\}_{r_k \le i < m_k}$ forms a basis of $\operatorname{im}(\partial_k) \subseteq \mathbb{F}^{m_{k-1}}$. But by definition $\{\partial_k b_i^{(k)}\}_{r_k \le i < m_k} = \{b_i^{(k-1)}\}_{0 \le i < r_{k-1}}$ and any linearly independent subset can be extended to a basis of $\mathbb{F}^{m_k}$. □

Remark 4.12. When $\mathbb{F} \subseteq \mathbb{C}$ we might realize line 6 of algorithm 5 by computing a basis over $\mathbb{F}$ of the kernel of the adjoint (conjugate transpose) of the matrix

$$(b_0^{(k-1)}, \dots, b_{r_{k-1}-1}^{(k-1)}, 0, \dots, 0) \in \operatorname{Mat}_{m_{k-1}, m_{k-1}}(\mathbb{F}).$$

This suggests that we could also give an algorithm for combinatorial torsion that only talks about matrices and not about vectors.

Algorithm 6 Torsion of a bounded, f.g., based acyclic chain complex over $\mathbb{F} \subseteq \mathbb{C}$

Require: $\{D^{(k)}\}_{1 \le k \le d}$.
$D^{(k)} \in \operatorname{Mat}_{m_{k-1}, m_k}(\mathbb{F})$ represents the boundary operator $C_k \to C_{k-1}$ of a chain complex C over $\mathbb{F} \subseteq \mathbb{C}$ with respect to a distinguished basis. C is assumed to be f.g., **exact** and bounded by 0 and $d \in \mathbb{N}$.
Ensure: $\tau \in \mathbb{F}^*$ the combinatorial torsion of C.

```
Define $r_d := 0$ and $r_{d-1} := m_d$.
Define $K^{(d)} \in \operatorname{Mat}_{m_d, m_d}(\mathbb{F})$ to be the identity matrix.
for $k = d, d-2, \dots, 1$ do
    Define $r_{k-2} := m_{k-1} - r_{k-1} \in \mathbb{Z}$.
    Solve $(D^{(k)} K^{(k)})^\dagger \cdot K^{(k-1)} = 0 \in \operatorname{Mat}_{r_{k-1}, r_{k-2}}(\mathbb{F})$ for $K^{(k-1)} \in \operatorname{Mat}_{m_{k-1}, r_{k-2}}(\mathbb{F})$ such that $K^{(k-1)}$ has full rank.
return $\tau := \prod_{k=1}^{d} \det(D^{(k)} K^{(k)} \mid K^{(k-1)})^{(-1)^k}$.
```

Proof of algorithm 6. To see that algorithm 6 yields the correct number, we compare it to algorithm 5. The translation goes as follows: The $\{b_i^{(k-1)}\}_{0 \le i < r_{k-1}}$ correspond to the columns of the matrix $D^{(k)}K^{(k)}$. By construction and dimension reasons, the columns of $K^{(k-1)}$ span the orthogonal complement of the image of $D^{(k)}K^{(k)}$. So the columns of $K^{(k-1)}$ correspond to the $\{b_i^{(k-1)}\}_{r_{k-1}\leq i<m_{k-1}}$, which explain how the formula for τ in algorithm 6 arises from algorithm 5. □

Remark 4.13. The advantage of algorithm 6 is that, if we use a matrix decomposition of $D^{(k)}K^{(k)}$ to solve for $K^{(k-1)}$, we can later use the same matrix decomposition to compute the determinant. Say we have computed a LU-decomposition $D^{(k)}K^{(k)} = PL(\frac{U}{0})$ with $P \in \mathrm{Mat}_{m_{k-1},m_{k-1}}(\mathbb{F})$ a square permutation matrix, $L \in \mathrm{Mat}_{m_{k-1},m_{k-1}}(\mathbb{F})$ a lower triangular matrix with every diagonal entry equal to 1 and $U \in \mathrm{Mat}_{r_{k-1},r_{k-1}}(\mathbb{F})$ an upper diagonal matrix, such that $(\frac{U}{0}) \in \mathrm{Mat}_{m_{k-1},r_{k-1}}(\mathbb{F})$. Then $(D^{(k)}K^{(k)})^\dagger = (U^\dagger|0)L^\dagger P^T$ and we see that $K^{(k-1)}$ is given by the last r_{k-2}-columns of $P(L^\dagger)^{-1}$, i.e. $K^{(k-1)} = P(L^\dagger)^{-1}\left(\frac{0}{I_{r_{k-2}}}\right)$, where $I_{r_{k-2}} \in \mathrm{Mat}_{r_{k-2},r_{k-2}}(\mathbb{F})$ is the identity matrix and $\left(\frac{0}{I_{r_{k-2}}}\right) \in \mathrm{Mat}_{m_{k-1},r_{k-2}}(\mathbb{F})$. Further,

$$\begin{aligned}
\det(D^{(k)}K^{(k)} \mid K^{(k-1)}) &= \det\left(PL\left(\frac{U}{0}\right) \mid P(L^\dagger)^{-1}\left(\frac{0}{I_{r_{k-2}}}\right)\right)\\
&= \mathrm{sgn}(P)\det\left(L\left(\frac{U}{0}\right) \mid (L^\dagger)^{-1}\left(\frac{0}{I_{r_{k-2}}}\right)\right)\\
&= \mathrm{sgn}(P)\det\left(L\left(\begin{array}{c|c} U & 0\\ \hline 0 & 0\end{array}\right) + (L^{-1})^\dagger\left(\begin{array}{c|c} 0 & 0\\ \hline 0 & I_{r_{k-2}}\end{array}\right)\right)
\end{aligned}$$

The last formula would simplify a lot if L was unitary. This suggest that we might use singular value decomposition for computing torsion of exact bounded f.g. chain complexes over $\mathbb{C}$.

Proof of algorithm 7. We extend the recursive definition of r_k in algorithm 7 by setting $r_k = 0$ for all $k \geq d$ and prove the following statement by induction on $k \in \mathbb{Z}$:

$$\text{If } C \text{ is exact at } C_j \text{ for all } j > k, \text{ then } r_{i-1} = m_i - \mathrm{rk}(D^{(i+1)}) \text{ for all } i \geq k. \tag{15}$$

We start the induction by $k = d$ and the induction step will go from k to $k-1$. The case, $k = d$ is clear. Now assume the statement holds for $k+1$ and that C is exact at C_j, for all $j > k$. By induction hypothesis $r_{i-1} = m_i - \mathrm{rk}(D^{(i+1)})$ for all $i > k$ and by definition $r_{k-1} = m_k - r_k$. Plugging the former equation, for $i = k+1$, into the latter equation, yields

$$r_{k-1} = m_k - r_k = m_k - (m_{k+1} - \mathrm{rk}(D^{(k+2)})) = m_k - (m_{k+1} - \ker(D^{(k+1)}))$$

by exactness at $k+1$. Therefore, $r_{k-1} = m_k - \mathrm{rk}(D^{(k+1)})$ by rank-nullity, which concludes the induction step.

Since C is a complex, C is exact at C_k if and only if

$$\mathrm{rk}(D^{(k+1)}) \geq \dim\ker D^{(k)} = m_k - \mathrm{rk}(D^{(k)}) \iff \mathrm{rk}(D^{(k)}) \geq m_k - \mathrm{rk}(D^{(k+1)}).$$

Algorithm 7 Torsion of a bounded f.g. based chain complex over $\mathbb{C}$

```
Require: $\{D^{(k)}\}_{1\leq k\leq d}$.
         $D^{(k)} \in \mathrm{Mat}_{m_{k-1},m_k}(\mathbb{F})$ represents the boundary operator $C_k \to C_{k-1}$ of a
         chain complex $C$ over $\mathbb{C}$ with respect to a distinguished basis. $C$ is assumed
         to be f.g., exact and bounded by 0 and $d \in \mathbb{N}$.
Ensure: $\tau \in \mathbb{C}$ the torsion of $C$, which is non-zero if and only if $C$ is acyclic.

  Define $r_d := 0 \in \mathbb{Z}$; $r_{d-1} := m_d \in \mathbb{Z}$ and $\tau := 1 \in \mathbb{C}$.
  Define $K^{(d)} \in \mathrm{Mat}_{m_d,m_d}(\mathbb{C})$ to be the identity matrix $I_{m_d}$.
  for $k = d, d-2, \dots, 1$ do
      $r_{k-2} := m_{k-1} - r_{k-1} \in \mathbb{Z}$.
      Compute a singular value decompositon $D^{(k)} = U\Sigma V^\dagger$ with singular values $\zeta_i$,
      where zero singular values appear last in $\Sigma$.
      if $r_{k-1} > m_{k-1}$ or ($r_{k-1} > 0$ and $\zeta_{r_{k-1}} = 0$) then
          return $\tau = 0$.
      $W := (I_{r_{k-1}}|0)V^\dagger K^{(k)} \in \mathrm{Mat}_{r_{k-1},r_{k-1}}(\mathbb{C})$.
      Set $\tau = \tau \cdot \det(U)^{(-1)^k} \cdot \left(\prod_{i=1}^{r_{k-1}} \zeta_i\right)^{(-1)^k} \cdot \det(W)^{(-1)^k}$.
      $K^{(k-1)} := U \cdot \begin{pmatrix} 0 \\ I_{r_{k-2}} \end{pmatrix} \in \mathrm{Mat}_{m_{k-1},r_{k-2}}(\mathbb{C})$.
  if $r_{-1} = 0$, then return $\tau$; else return $\tau = 0$.
```

This is the case, if and only if the following holds:

$$m_k = \mathrm{rk}(D^{(k+1)}) \text{ or } \left(1 \leq m_k - \mathrm{rk}(D^{(k+1)}) \leq m_{k-1} \text{ and } \zeta_{m_k - \mathrm{rk}(D^{(k+1)})} \neq 0\right), \qquad (16)$$

recalling that the singular values ζ_i of $D^{(k)}$ are ordered.

Combining equation (15) and (16), yields that algorithm 7 returns 0, if C is not exact.

Now assume that C is acyclic and we prove that algorithm 7 returns the same number as algorithm 6. This would then conlcude the proof, since the torsion of an acyclic chain complex is, in particular, non zero. Equation 15, exactness of C and the rank-nullity theorem imply that $r_{i-1} = \mathrm{rk}(D^{(i)})$ for all $0 \leq i \leq d$.
Fix some $0 < k \leq d$ and consider some singular value decomposition $D^{(k)} = U\Sigma V^\dagger$ with $U \in U(m_{k-1})$, $\Sigma = (\zeta_1, \dots, \zeta_{r_{k-1}}, 0, \dots, 0) \in \mathrm{Mat}_{m_{k-1},m_k}(\mathbb{C})$, $V \in U(m_k)$. Then $(D^{(k)}K^{(k)})^\dagger = (K^{(k)})^\dagger V^\dagger \Sigma^T U^\dagger$. The matrix

$$K^{(k-1)} := U \left(\frac{0}{I_{r_{k-2}}}\right) \in \mathrm{Mat}_{m_{k-1},r_{k-2}}(\mathbb{C})$$

with $\left(\frac{0}{I_{r_{k-2}}}\right) \in \mathrm{Mat}_{m_{k-1},r_{k-2}}(\mathbb{C})$ has orthonormal columns and satisfies

$$(D^{(k)}K^{(k)})^\dagger \cdot K^{(k-1)} = 0 \in \mathrm{Mat}_{r_{k-1},r_{k-2}}(\mathbb{F}).$$

So, we may assume that we have picked this specific $K^{(k-1)}$ in algorithm 6. Further, we can write

$$\Sigma = \begin{pmatrix} \operatorname{diag}(\zeta_1, \dots, \zeta_{r_{k-1}}) & 0 \\ 0 & 0 \end{pmatrix} \in \operatorname{Mat}_{m_{k-1}, m_k}(\mathbb{C}). \tag{17}$$

We may write $\left(\frac{W}{W'}\right) = V^\dagger K^{(k)} \in \operatorname{Mat}_{m_k, r_{k-1}}(\mathbb{C})$ for $W \in \operatorname{Mat}_{r_{k-1}, r_{k-1}}(\mathbb{C})$ and $W' \in \operatorname{Mat}_{r_k, r_{k-1}}(\mathbb{C})$. We obtain that

$$\Sigma V^\dagger K^{(k)} = \begin{pmatrix} \operatorname{diag}(\zeta_1, \dots, \zeta_{r_{k-1}}) W \\ 0 \end{pmatrix} \in \operatorname{Mat}_{m_{k-1}, r_{k-1}}(\mathbb{C}).$$

Hence,

$$\begin{aligned}
\det(D^{(k)} K^{(k)} \big| K^{(k-1)}) &= \det\left(U\Sigma V^\dagger K^{(k)} \;\Big|\; U \left(\frac{0}{I_{r_{k-2}}} \right) \right) = \det(U) \det\left(\Sigma V^\dagger K^{(k)} \;\Big|\; \left(\frac{0}{I_{r_{k-2}}} \right) \right) \\
&= \det(U) \det \begin{pmatrix} \operatorname{diag}(\zeta_1, \dots, \zeta_{r_{k-1}}) W & 0 \\ 0 & I_{r_{k-2}} \end{pmatrix} \\
&= \det(U) \left(\prod_{i=1}^{r_{k-1}} \zeta_i \right) \det(W),
\end{aligned}$$

which was what we wanted to prove. □

Remark 4.14 (Remarks on an implementation of algorithm 7). We don't need to safe the $K^{(k)}$'s in algorithm 7, so we could overwrite them in every step of the for-loop.
Also, we might want to check whether $r_{k-1} > m_{k-1}$ in the beginning of the loop, which would safe us to compute one more singular value decomposition, when we will return 0 anyhow.
If $r_{k-1} = 0$ for some k and C_j is exact for all $j > k$, then we see from the proof of algorithm 7 that $D^{(k+1)}$ is surjective. Because C is a complex, then $D^{(k)} = 0$ and $C^{(k)}$ will be exact at k. One possible singular value decomposition of $D^{(k)}$ is then given by $U = I_{m_{k-1}}$ the identity matrix, $\Sigma \in \operatorname{Mat}_{m_{k-1}, m_{k-2}}(\mathbb{C})$ the zero matrix and $V^\dagger = I_{m_k}$ the identity matrix. Now $W \in \operatorname{Mat}_{0,0}(\mathbb{C})$ must have determinant 1, and $\prod_{i=1}^{r_{k-1}} \zeta_i = 1$ is the empty product. Further, $r_{k-2} = m_{k-1}$ and $K^{(k-1)} = U = I_{m_{k-1}}$. All in all, it makes sense to check in the beginning of the for-loop in algorithm 7, whether $r_{k-1} = 0$. In this case, we don't need to compute a singular value decomposition of $D^{(k)}$ but could instead set $K^{(k-1)} = I_{m_{k-1}}$, $r_{k-2} = m_{k-1}$ and leave τ unaltered. Then we could continue the for-loop with $k - 1$.

Remark 4.15 (Algorithm 7 also works for $\mathbb{F} \subsetneq \mathbb{C}$). If $\mathbb{F} \subsetneq \mathbb{C}$ then algorithm 7 still gives the correct number. This is because we might view the chain complex as a chain complex over $\mathbb{C}$ from the beginning and, as we know that $\tau(C)$ doesn't depend on the chosen basis, it does not matter if the coefficients have stayed in $\mathbb{F}$ all the time or not. Further, tensoring a finitely generated free chain complex over $\mathbb{F}$ with $\mathbb{C}$ preserves and reflect exactness, so we might check exactness over $\mathbb{F}$ or $\mathbb{C}$ and obtain the same result.
If $\mathbb{F} \subseteq \mathbb{R}$, we could force algorithm 7 to only use real matrices by doing real valued singular value composition.

4.3. Reidemeister torsion

Definition 4.16 (Reidemeister Torsion)**.** Let X be a finite, connected simplicial set with universal cover Z and fundamental group G, which we identify with the deck transformation group of the covering projection $Z \to X$. Let $\tilde{\partial}$ be the boundary map of the left-$\mathbb{Z}[G]$-module chain complex $C^{\mathbf{CW}}(Z)$.
Suppose for all $k \in \mathbb{Z}$ the boundary map $\tilde{\partial}_k : C_k^{\mathbf{CW}}(Z) \to C_{k-1}^{\mathbf{CW}}(Z)$ is represented by a matrix $M^{(k)}$ with coefficients in $\mathbb{Z}[G]$ with respect to the basis' X_k^{nd} and X_{k-1}^{nd} for $C_k^{\mathbf{CW}}(Z)$ and $C_{k-1}^{\mathbf{CW}}(Z)$, respectively.
Let $\phi : G \to \mathrm{GL}(\mathbb{C}^n)$ be a n-dimensional complex representation of G which induces a ring homomorphism $\tilde{\phi} : \mathbb{Z}[G] \to \mathrm{End}(\mathbb{C}^n)$ from the integral group ring over G to the endomorphism ring of $\mathbb{C}^n$.
Define the *twisted (normalized) chain complex* $C^{\mathbf{CW}}(X, \phi)$ with k-th chain group

$$\text{given by the } \mathbb{C}\text{-vector space } C_k^{\mathbf{CW}}(X,\phi) = \bigoplus_{\sigma \in X_k^{\mathrm{nd}}} \sigma \cdot \mathbb{C}^n \quad \text{for all } 0 \le k \le \dim(X) \quad (18)$$

and k-th boundary map

$$\partial_k^{\phi} : C_k^{\mathbf{CW}}(X,\phi) \to C_{k-1}^{\mathbf{CW}}(X,\phi) : \quad \sigma_j \cdot v \mapsto \sum_{\sigma_i \in X_{n-1}^{nd}} \sigma_i \cdot \tilde{\phi}(M_{i,j}^{(k)})(v). \qquad (19)$$

The chain complex $C^{\mathbf{CW}}(X, \phi)$ can be based by the standard basis as suggested in equation (18). Further, $C^{\mathbf{CW}}(X, \phi)$ is finitely generated and bounded. So, we might define the *Reidemeister torsion*

$$\tau(X, \phi) := \tau(C^{\mathbf{CW}}(X, \phi)) \in \mathbb{C}^*/(\pm\mathrm{im}(\det \circ \phi)) \cup \{0\},$$

as the combinatorial torsion of $C^{\mathbf{CW}}(X, \phi)$, which we defined in definition 4.10, but we will identify nonzero values that differ by multiplication with elements in $\pm\mathrm{im}(\det \circ\phi)$. Note that by definition $\tau(X, \phi) = 0$ if and only if $C^{\mathbf{CW}}(X, \phi)$ is not acyclic.

Remark 4.17. The twisted chain complex $C^{\mathbf{CW}}(X, \phi)$ is isomorphic to $C^{\mathbf{CW}}(Z) \otimes_{\mathbb{Z}[G]} \mathbb{C}^n$, considered as a right $\mathbb{C}$-module. Here $\mathbb{C}^n$ becomes a left $\mathbb{Z}[G]$-module via the ring homomorphism $\tilde{\phi} : \mathbb{Z}[G] \to \mathrm{End}(\mathbb{C}^n)$ and the left $\mathrm{End}(\mathbb{C}^n)$-module structure on $\mathbb{C}^n$.

Remark 4.18. Equation (19) is saying that the matrix of ∂_k^{ϕ} with respect to the distinguished basis of $C^{\mathbf{CW}}(X, \phi)$ is obtained from the matrix $M^{(k)}$ by replacing every entry $M_{i,j}^{(k)} = \sum_{g \in G} a_g g$, ($a_g \in \mathbb{Z}, a_g = 0$ for all but finite g), with the matrix $\sum_{g\in G} a_g \phi(g)$.
For example, if $G = \{1, \gamma\} = \mathbb{Z}_2$, $\phi(\gamma) = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$ and $M^{(k)} = \begin{pmatrix} 1-\gamma & -1 \\ 0 & \gamma \end{pmatrix}$, then ∂_k^{ϕ} is represented by the block matrix

$$\left(\begin{array}{cc|cc} 2 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 \\ \hline \multicolumn{2}{c|}{\multirow{2}{*}{$\mathbf{0}$}} & -1 & 0 \\ & & 0 & -1 \end{array}\right) \in \mathrm{Mat}_{4,4}(\mathbb{C}).$$

Lemma 4.19. *$\tau(X, \phi)$ is independent of the choice of lifts $\sigma \in X_k^{nd}$ that determine the $\mathbb{Z}[G]$-basis of $C_k^{\mathbf{CW}}(Z)$.*

Proof. Note that if we change the $\mathbb{Z}[G]$-basis of $C_k^{\mathbf{CW}}(Z)$ by replacing a lift $\tilde{\sigma}$ of $\sigma \in X_k^{\mathrm{nd}}$ with $g(\tilde{\sigma})$ for some deck-transformation $g \in G$, then this corresponds to multiplying $\tau(X, \phi)$ by $\det(\phi(g))^{(-1)^k}$. But we explicitly defined $\tau(X, \phi)$ only up to such factors. □

Remark 4.20. Let X be a finite simplicial set with fundamental group G and Euler characteristic $\chi(X)$. For any n-dimensional representation $\phi : G \to \mathrm{GL}_n(\mathbb{C})$ of G we have that the Euler characteristic of the twisted chain complex $C^{\mathbf{CW}}(X, \phi)$ is $n \cdot \chi(X)$. So, $C^{\mathbf{CW}}(X, \phi)$ can only be acyclic if $\chi(X)$ is zero. Hence Reidemeister torsion is only interesting for simplicial sets X with $\chi(X) = 0$. This is the case, for example, when $|X|$ is homotopy equivalent to a odd-dimensional manifold, see [8, cor. 3.37].

Proposition 4.21. *The Reidemeister torsion $\tau(X, \phi)$ is independent of the simple-homotopy type of $|X|$. So, by [2], $\tau(X, \phi)$ is, in particular, independent of the homeomorphism type of $|X|$, which, in turn, implies that $\tau(X, \phi)$ is independent of the isomorphism type of X by proposition 2.26.*

To prove the previous proposition we assume some familiarity with torsion of chain complexes C, denoted $\tau(C)$, over arbitrary rings and simple-homotopy theory as discussed in [11], [3] and [20]. We will write $K_1(R)$ and $\mathrm{WH}(G)$ multiplicatively for any ring R and group G.

Lemma 4.22. *[11, lem. 1.10] Let $n, m \geq 0$ an R a ring. Let $A \in GL_m(End(R^n))$. Let $B \in GL_{nm}(R)$ be the matrix where the $(i,j)^{th}$-block represents $A_{i,j}$ with respect to the standard basis of R^n. Then the assignment $A \mapsto B$ induces a group isomorphism $h : K_1(End(R^n)) \to K_1(R)$.*

Lemma 4.23. *Let $n \geq 0$, R a ring, and C an chain complex of $End(R^n)$-modules. Assume that $C_k = 0$ for $k < 0$ and $k > d$ and, moreover, that C_k has basis $\{e_i^{(k)}\}_{0 \leq i < m_k}$ for all $0 \leq k \leq d$.*
Define $C' := C \otimes_{End(R^n)} R^n$ as a chain complex over R, where we consider R^n as a left $End(R^n)$-module via matrix multiplication. Then C' is bounded and for all $0 \leq k \leq d$ the $\mathbb{C}$-vector space C_k is based by $\{e_i^{(k)} \otimes f_s\}_{\substack{0 \leq i \leq m_k, \\ s=1,\ldots,n}}$, where $\{f_1, \ldots, f_n\}$ denotes the standard basis of R^n.
Further, C' is acyclic if and only if C is acyclic and in that case

$$\tau(C') = h(\tau(C)) \in K_1(R)$$

for h the isomorphism from lemma 4.22.

Proof. If C is acyclic, then C admits a chain contraction using that C is free and bounded. Hence, also C' admits a chain contraction and is, in particular, acyclic.
Now assume that C' is acyclic and take $c \in \ker(\partial_k) \subseteq C_{k-1}$. Then for all $s = 1, \ldots, n$

there exist $b_s \in C'_{k+1}$ with $\partial_k(b_s) = c \otimes f_s$, i.e. $\partial_k(\sum_{s=1}^n b_s) = \sum_{s=1}^n c \otimes f_s$. We may write $b_s = \sum_{0\le i\le m_k} e_i \otimes v_{i,s}$ for some $v_{i,s} \in \mathbb{C}^n$ for all $s = 1, \dots, n$. Define $M_{s,i} \in \operatorname{End}(R^n)$ by

$$M_{s,i}(f_l) = \begin{cases} v_{i,s}, & \text{if } l = s, \\ 0, & \text{else,} \end{cases}$$

and set $\tilde{b}_s := \sum_{0\le i<m_k} M_{s,i} e_i \in C_k$. Then, for $0 \le l \le n$,

$$\partial_k(\tilde{b}_s) \otimes f_l = \partial_k(\tilde{b}_s \otimes f_l) = \partial_k\left(\sum_{0\le i<m_k} e_i \otimes M_{s,i} f_l\right) = \begin{cases} \partial_k(b_s) = c \otimes f_s, & \text{if } l = s, \\ \partial_k(0) = 0, & \text{else.} \end{cases}$$

Hence, for $b := \sum_{s=1}^n \tilde{b}_s$ we have $\partial_k(b) \otimes f_s = c \otimes f_s$ for all $s = 1, \dots, n$. Say $a := \partial_k(b) - c = \sum_{0\le i<m_{k-1}} a_i e_i$ for some $a_i \in \operatorname{End}(R^n)$. Then for all $s = 1, \dots, n$ and $i \in I_{k-1}$, we have

$$0 = \partial_k(b) \otimes f_s - c \otimes f = a \otimes f_s = \sum_{0\le i<m_{k-1}} e_i \otimes a_i(f_s),$$

so $a_i(f_s) = 0$ for all $0 \le i < m_{k-1}$. Since $\{f_s\}_{s=1,\dots,n}$ forms a generating system of R^n, we obtain $a_i = 0$ for $0 \le i < m_{k-1}$, which proves $0 = a = c - \partial_k(b)$. In other words, c is a boundary. As c was an arbitrary cycle, we obtain that C is acyclic.

Now assume C is acyclic. By [3, §14.] there exists a finitely generated, based $\operatorname{End}(R^n)$-module F such that $\operatorname{im}(\partial_k \oplus \operatorname{id}_F) \subseteq C_{k-1} \otimes F$ is free for all $0 \le k \le d$ and such that $\tau(C \oplus F) = \tau(C)$, as well as, $\tau(C') = \tau((C \oplus F) \otimes R^n)$. So, we may replace C with $C \oplus F$ and assume that $\operatorname{im}(\partial_k)$ is free for all $0 \le k \le d$.

Because

$$0 \to \operatorname{im}(\partial_{k+1}) \to C_k \xrightarrow{\partial_k} \operatorname{im}(\partial_k) \to 0$$

is split exact, for all $1 \le k \le d+1$, there exists $\{b_i^{(k-1)}\}_{0\le i<r_{k-1}}$ in C_{k-1} forming a basis of $\operatorname{im}(\partial_k) \subseteq C_{k-1}$ and there are $\{b_{i+r_k}^{(k)}\}_{0\le i<r_{k-1}}$ with $\partial_k b_{i+r_k}^{(k)} = b_i^{(k-1)}$ for $0 \le i < r_{k-1}$ such that $\{b_i^{(k)}\}_{0\le i<m_k}$ is a basis of C_k.

For $0 \le k \le d$ let $A^{(k)} \in \operatorname{Mat}_{m_k,m_k}(\operatorname{End}(\mathbb{C}^n))$ satisfy $b_j^{(k)} = \sum_{0\le i<m_k} A_{i,j}^{(k)} e_i^{(k)}$ for all $0 \le j < m_k$. Then by [3, §16.1]

$$\tau(C) = \prod_{i=0}^{d} [A^{(k)}]^{(-1)^{k+1}}.$$

For all $0 \le k \le d$, the collection $\{b_i^{(k-1)} \otimes f_s\}_{\substack{0\le i\le r_{k-1},\\ s=1,\dots,n}}$ forms a R-basis of $\operatorname{im}(\partial_k^\phi) \subseteq C'_{k-1}$ and $\partial'_k(b_{i+r_k}^k \otimes f_s) = b_i^{(k-1)} \otimes f_s$ for all $0 \le i < r_{k-1}$ and $1 \le i \le n$.

Define $B^{(k)} \in \operatorname{Mat}_{n\cdot m_k, n\cdot m_k}(\mathbb{C})$ to consist of m_k^2 blocks of $n \times n$-matrices, where the (i,j)th-block is the matrix representing $A_{i,j}^{(k)}$ with respect to the standard basis. Then for all $0 \le k \le d$

$$\sum_{\substack{0\le i<m_k,\\ s=1,\dots,n}} B_{n\cdot i+s, n\cdot j+l}^{(k)} (e_i^{(k)} \otimes f_s) = \sum_{0\le i<m_k} e_i^{(k)} \otimes A_{i,j}^{(k)}(f_l) = \sum_{0\le i<m_k} A_{i,j}^{(k)} e_i^{(k)} \otimes f_l = b_j^{(k)} \otimes f_l$$

for all $0 \leq i < m_k$ and $s = 1, \ldots, n$. So, by [3, §16.1] applied to C', we have

$$\tau(C') = \prod_{i=0}^{d} [B^{(k)}]^{(-1)^{k+1}},$$

and by the definition of h in lemma 4.22 we have $h([A^k_{i,j}]) = [B^{(k)}]$, which concludes the prove of this lemma. □

Proposition 4.24 (Invariance of Reidemeister torsion)**.** *Let X, Y be finite connected simplicial sets. Let $Z, W \in$ **sSet** be universal covers of X and Y, respectively. Let G and H be the fundamental groups of X and Y, respectively, which we identify with the deck-transformation groups of $Z \to X$ and $W \to Y$, respectively. Let $\phi : H \to GL(\mathbb{C}^n)$ be a homomorphism. Define $\psi := \phi \circ \pi_1(f) : G \to GL(\mathbb{C}^n)$. Let $f : |X| \to |Y|$ be a homotopy equivalence and $\tau(f) \in WH(H)$ be the Whitehead torsion of f. Then*

$$\tau(Y, \phi) = \tau(X, \psi) \cdot (\det \circ \tilde{h} \circ \tilde{\phi}_*)(\tau(f)) \ \in \mathbb{C}^*/(\pm \det \circ \phi(H))) \cup \{0\},$$

where

- *$\tilde{\phi}_*$ is the group homomorphism $WH(H) \to K_1(End(\mathbb{C}^n))/ \pm \phi(G)$ induced by $\phi : H \to GL(\mathbb{C}^n)$*
- *$\tilde{h} : K_1(End(\mathbb{C}^n))/ \pm \phi(G) \to K_1(\mathbb{C})/ \pm \phi(G)$ is the group isomorphism induced by h from lemma 4.22*
- *$\det : K_1(\mathbb{C})/ \pm \phi(G) \to \mathbb{C}^*/ \pm \phi(G)$ is the group isomorphism induced by the determinant.*

Proof. The group homomorphisms $\phi : H \to \mathrm{GL}(\mathbb{C})$ and $\psi : G \to \mathrm{GL}(\mathbb{C}^n)$ induce ring homomorphisms $\tilde{\phi} : \mathbb{Z}[H] \to \mathrm{End}(\mathbb{C}^n)$ and $\tilde{\psi} : \mathbb{Z}[G] \to \mathrm{End}(\mathbb{C}^n)$. By lemma 4.23 the chain complexes $C^{\mathbf{CW}}(Y, \tilde{\phi}) := C^{\mathbf{CW}}(W) \otimes_{\mathbb{Z}[G]} \mathrm{End}(\mathbb{C}^n)$ and $C^{\mathbf{CW}}(X, \tilde{\psi}) := C^{\mathbf{CW}}(Z) \otimes_{\mathbb{Z}[G]} \mathrm{End}(\mathbb{C}^n)$ are acyclic if and only if $C^{\mathbf{CW}}(Y, \phi)$ and $C^{\mathbf{CW}}(X, \psi)$ are acyclic, respectively. By [20, thm. 9.1], $C^{\mathbf{CW}}(Y, \tilde{\phi})$ being acyclic, implies the same for $C^{\mathbf{CW}}(X, \tilde{\psi})$. The converse direction holds by applying [20, thm. 9.1] to a homotopy inverse of f. Hence, $C^{\mathbf{CW}}(X, \psi)$ is acyclic if and only if $C^{\mathbf{CW}}(Y, \phi)$ is acyclic. We now assume that this holds. By [20, thm. 9.1], we have

$$\tau(C^{\mathbf{CW}}(Y, \tilde{\phi})) = \tau(C^{\mathbf{CW}}(X, \tilde{\psi})) \cdot \tilde{\phi}_* \tau(f) \in K_1(\mathrm{End}(\mathbb{C}^n)/ \pm \phi(H). \tag{20}$$

Applying $\tilde{h}$ to equation (20) and then using lemma 4.23 yields that

$$\begin{aligned} \tau(C^{\mathbf{CW}}(Y, \phi)) &\overset{4.23}{=} \tilde{h}(\tau(C^{\mathbf{CW}}(Y, \tilde{\phi}))) \overset{(20)}{=} \tilde{h}(\tau(C^{\mathbf{CW}}(X, \psi))) \cdot \tilde{h}(\phi_* \tau(f)) \\ &\overset{4.23}{=} \tau(C^{\mathbf{CW}}(X, \psi)) \cdot \tilde{h}(\phi_* \tau(f)) \in K_1(\mathbb{C})/ \pm \phi(H) \end{aligned} \tag{21}$$

and now the result follows by applying the isomorphism induced by the determinant to equation (21). □

Proof of proposition 4.21. This follows from proposition 4.24 because simple homotopy equivalence satisfy $\tau(f) = 1$. □

Lemma 4.25. *Let $\phi : G \to \mathbb{C}^n, \psi : G \to \mathbb{C}^m$ be representations of the fundamental group G of a finite, connected simplicial set X. Then*

$$\tau(X, \phi \oplus \psi) = \tau(X, \phi) \cdot \tau(X, \psi) \mod \det \phi(G) \cdot \det \psi(G).$$

Moreover, if ϕ and ψ are equivalent, then $\tau(X, \phi) = \tau(X, \psi)$.

Proof. Using the notations of definition 4.16, the assignment

$$C^{\mathbf{CW}}(X, \phi) \oplus C^{\mathbf{CW}}(X, \psi) \to C^{\mathbf{CW}}(X, \phi \oplus \psi),\ (\sigma_j \cdot v) \oplus (\sigma_l \cdot w) \mapsto \sigma_j \cdot (v \oplus 0) + \sigma_l \cdot (0 \oplus w)$$

induces an isomorphism of chain-complexes that preserves the distinguished basis. So, $\tau(C^{\mathbf{CW}}(X, \phi \oplus \psi)) = \pm\tau(C^{\mathbf{CW}}(X, \phi)) \cdot \tau(C^{\mathbf{CW}}(X, \psi))$ by [20, lem. 3.4], which proves the first statement.

Now suppose there is $A : \mathbb{C}^n \to \mathbb{C}^n$ invertible with $A \circ (\phi(g)) = (\psi(g)) \circ A : \mathbb{C}^n \to \mathbb{C}^n$ for all $g \in G$. Then, using the notations of definition 4.16, the assignment

$$C^{\mathbf{CW}}(X, \phi) \to C^{\mathbf{CW}}(X, \psi),\ \sigma \cdot v \mapsto \sigma \cdot Av$$

induces an isomorphism h of chain complexes. This already proves the statement in the case that $C^{\mathbf{CW}}(X, \phi)$ is not acyclic. Now let us assume that $C^{\mathbf{CW}}(X, \phi)$ is exact. The image of the distinguished basis of $C_k^{\mathbf{CW}}(X, \phi)$ under h_k is obtained from the distinguished basis of $C_k^{\mathbf{CW}}(X, \psi)$ by applying the block matrix $\operatorname{diag}(A, \dots, A)$. So, unraveling definition 4.10,

$$\tau(C^{\mathbf{CW}}(X, \phi)) = \prod_{k=0}^{\dim(X)} (\det(A)^{\# X_k^{\mathrm{nd}}})^{(-1)^{k+1}} \cdot \tau(C^{\mathbf{CW}}(X, \psi)).$$

Because X has Euler characteristic zero, we obtain

$$\prod_{k=0}^{\dim(X)} \det(A)^{(-1)^{k+1}} = \det(A)^{(\sum_{k=0}^{dim(X)} (-1)^{k+1} \# X_k^{\mathrm{nd}})} = \det(A)^0 = 1,$$

which proves the statement.

A more conceptual proof of the second statement of the lemma is the following:
We might identify $\tau(X, \phi)$ with the torsion of $C^{\mathbf{CW}}(X, \tilde{\phi}) := C^{\mathbf{CW}}(Z) \otimes_{\mathbb{Z}[G]} \operatorname{End}(\mathbb{C}^n)$ in $K_1(\operatorname{End}\mathbb{C}^n))/\phi(G)$, as we did in the proof of proposition 4.24. Here we consider $\operatorname{End}(\mathbb{C}^n)$ as a left $\mathbb{Z}[G]$-module via the ring-homomorphism $\tilde{\psi} : \mathbb{Z}[G] \to \operatorname{End}(\mathbb{C}^n)$ induced by ψ. Also, $C^{\mathbf{CW}}(X, \tilde{\phi})$ is considered as a chain-complex of right $\operatorname{End}(\mathbb{C}^n)$-modules.

Suppose ψ and ϕ are equivalent. Then, there exists a linear isomorphism $A : \mathbb{C}^n \to \mathbb{C}^n$ such that $(\psi(g)) \circ A = A \circ (\phi(g))$ for all $g \in G$. Now

$$h : C^{\mathbf{CW}}(X, \tilde{\phi}) \to C^{\mathbf{CW}}(X, \tilde{\psi}), \quad \sigma \otimes M \mapsto \sigma \otimes A \circ M$$

defines an isomorphism of chain complexes of $\mathbb{Z}[G]$-modules which preserves the distinguished basis. h also gives an isomorphism between $C^{\mathbf{CW}}(X, \tilde{\phi})$ and $C^{\mathbf{CW}}(X, \tilde{\psi})$ as chain complexes of right $\mathrm{End}(\mathbb{C}^n)$-modules. Since h preserves the distinguished basis, we have $\tau(C^{\mathbf{CW}}(X, \tilde{\phi})) = \tau(C^{\mathbf{CW}}(X, \tilde{\psi})) \in K_1(\mathrm{End}(\mathbb{C}^n))$ and the result follows. □

Remark 4.26. $\tau(X, \phi)$ equals ± 1 for every rational representation of a finite fundamental group $\pi_1(X)$ because every such rational representation is equivalent to a representation by integer matrices, which have determinant ± 1.

Remark 4.27. Let X be a finite, connected simplicial set with **finite** fundamental group G. Let $\phi : G \to \mathbb{C}^n$ be a presentation of G. Then $|\det \phi(g)| = 1$ for all $g \in G$, because g, and hence, $\det \phi(g) \in \mathbb{C}^*$ has finite order. Therefore, the absolute value $|\tau(X, \phi)|$ is always a well-defined non-negative real number. In the case, that ϕ is a real representation, $\tau(X, \phi)$ is defined up to a sign.

Moreover, by complete irreducibility, ϕ is equivalent to a direct sum of irreducible complex representations $\psi_1, \ldots, \psi_r$ of G, i.e. $\phi \sim \psi_1 \oplus \ldots \psi_r$. By lemma 4.25 we have $|\tau(X, \phi)| = \prod_{i=1}^{r} |\tau(X, \psi_i)|$. So, if we are only interested in orthogonal representations, or, if we are only interested in the absolute values of torsions, it suffices to compute $\tau(X, \phi)$ for all irreducible representations of G.

Remark 4.28 (Application of Reidemeister torsion). Let us use the notation of proposition 4.24 and assume that H is finite and that $C^{\mathbf{CW}}(Y, \psi)$ is acyclic for all non-trivial irreducible representations of H.

By [11, thm. 6.9] an element $\omega \in \mathrm{WH}(H)$ has finite order if and only if $\tilde{\phi}_*(\omega) = 1$ for every irreducible orthogonal representation ϕ of H. For ϕ the trivial representation, $\tilde{\phi}_* \tau(f) = 1$ by remark 4.26. So, using proposition 4.24, and [11, thm. 6.9], we obtain that $\tau(f)$ has finite order in $\mathrm{Wh}(H)$, if and only if $\tau(Y, \phi) = \tau(X, \psi)$ for all irreducible orthogonal representations ψ of H.

This result can be strengthened in the case that H is, moreover, abelian. By [11, thm. 6.4], $\mathrm{WH}(H)$ is a free abelian group of finite rank. We obtain that f is a simple-homotopy equivalence if and only if $|\tau(Y, \phi)|$ and $|\tau(X, \phi \circ \pi_1(f))|$ are equal non-negative real numbers for every orthogonal representations ϕ of H.

Proof of algorithm 8. Let us denote the fundamental group of X with G and let $Z \to X$ be a universal covering projection. We identify G with the deck-transformation group of the covering projection $Z \to X$.
Algorithm 3 applied to $X, \{e_i\}_{0 \leq i < m}, \{n_i\} = \emptyset$ computes $C^{\mathbf{CW}}(Z)$ by giving the matrices $M^{(k)}$ of the boundary operators with respect to the $\mathbb{Z}[G]$-basis of the chain groups used in the definition of Reidemeister torsion in 4.16. To obtain the twisted chain complex $C^{\mathbf{CW}}(X, \phi)$ we have to replace the matrices $M^{(k)}$ with matrices with coefficients in $\mathbb{C}$ by the recipe given in remark 4.18.

The given algorithm 8 does exactly this, but in one step, i.e. we do algorithm 3 but replace the matrix entries already when they are computed. □

Algorithm 8 Reidemeister torsion of simplicial set X for given representation of $\pi_1(X)$

Require: X: Connected simplicial set, $X_k^{\text{nd}} = \{\sigma_1^{(k)}, \dots, \sigma_{m_k}^{(k)}\}$ for all $k \leq \dim X < \infty$,
$\{e_i\}_{1\leq i\leq m}$: $X_1^{\text{nd}} \setminus \{e_1, \dots, e_m\}$ set of edges of a spanning tree Γ in $\text{sk}_1(X)$,
ψ: $\{e_1, \dots, e_m\} \to \text{GL}_n(\mathbb{C})$ inducing a homomorphism $\phi : \pi_1(X) \to \text{GL}_n(\mathbb{C})$.
(Here we identify $\pi_1(X)$ with a quotient of $F(e_1, \dots, e_m)$ via prop. 2.49.)

Ensure: τ: The Reidemeister torsion $\tau(X, \phi) \in \mathbb{C}$.
(Here $\tau = 0$ if and only if $C^{\mathbf{CW}}(X, \phi)$ is not exact.)

```
deckTrnsf := $\{k : [\,] \text{ for } k = 1, \dots, \dim(X)\}$        ▷ encodes the $g_0$'s from algorithm 2
twistComplx := $\{k : M_k := \text{matrix}(\mathbb{C}, n \cdot m_{k-1}, n \cdot m_k) \text{ for } k = 1, \dots, \dim X\}$
                                        ▷ These matrices are defined to be zero and will be filled later
for $i = 1, \dots, m_1$ do
    $\sigma_j^{(0)} := d_0\sigma_i^{(1)}$, $\sigma_l^{(0)} := d_1\sigma_i^{(1)}$        ▷ Obtain list indices j, l
    Set $M_1$.submatrix$(nl, ni, n \times n) = -I_n$ to be the identity matrix.
    if $\sigma_i^{(1)} = e_r$ for some $r = 1, \dots, m$ then
        deckTrnsf[1].append($\psi(e_j)$); $M_1$.submatrix$(nj, ni, n \times n) += \psi(e_j)$
    else
        deckTrnsf[1].append($I_n$); $M_1$.submatrix$(nj, ni, n \times n) += I_n$
for $k = 2, \dots, \dim X$ do
    for $i = 1, \dots, m_k$ do
        for $r = 1, \dots, k$ do
            if $d_r\sigma_i^{(k)} = \sigma_j^{(k-1)}$ is nondegenerate then
                $M_k$.submatrix$(nj, ni, n \times n) += (-1)^r I_n$
        $d_k\sigma_i^{(k)} = s_I(\sigma_j^{(s)})$        ▷ Obtain degeneracies, dimension and list index of $d_k\sigma_i^{(k)}$
        if $0 \in I$ then
            if $d_0(\sigma_i^{(k)}) = \sigma_l^{(k-1)}$ is nondegenerate then
                $M_k$.submatrix$(nl, ni, n \times n) += I_n$
            deckTrnsf[$k$].append(1)
        else
            $g_0$ := deckTrnsf[$s$][$j$]; deckTrnsf[$k$].append($g_0$)
            if $d_0(\sigma_i^{(k)}) = \sigma_l^{(k-1)}$ is nondegenerate then
                $M_k$.submatrix$(nl, ni, n \times n) += g_0$

Apply algorithm 7 to the chain complex twistComplx to obtain its torsion $\tau \in \mathbb{C}$.
return $\tau$
```

5. Implementation

All algorithms discussed in this paper have been implemented in [18, SageMath] and are available under `https://github.com/fNeugebauer/Computation-of-Combinatorial-Torsion` together with example computations. The source code with examples is also printed in the appendix of this bachelor thesis.

5.1. Implementation of simplicial sets

We use the implementation of finite simplicial sets that is provided by SageMath and documented in the reference manual [17].

One of our main examples, that we used to test our algorithms, are lens spaces. To construct simplicial sets with geometric realization a given lens space, we implemented a function in SageMath that computes joins of simplicial sets, as well as, a function that computes joins of morphisms of simplicial sets.

Let $p \in \mathbb{N}_{>0}$ be a positive integer and $q_1, \ldots, q_n \in \mathbb{Z}$ be integers such that $\gcd(p, q_i) = 1$ for all $i = 1, \ldots, n$. Fix a p^{th}-root of unity $\zeta \in \mathbb{C}^*$. Define $\rho_i : S^1 \to S^1, x \mapsto \zeta^{q_i} \cdot x$ for all $i = 1, \ldots, n$. Then the n-fold join of the morphisms ρ_i yields a continuous map $F : S^{2n-1} \to S^{2n-1}$ given by the composition

$$S^{2n-1} \cong S^1 \star \cdots \star S^1 \xrightarrow{\rho_1 \star \cdots \star \rho_n} S^1 \star \cdots \star S^1 \cong S^{2n-1}.$$

As $F^p = \mathrm{id}$, F generates an action of $\mathbb{Z}/(p)$ on S^{2n-1}. The quotient of S^{2n-1} by this action of $\mathbb{Z}/(p)$ yields the lens space $L(p; q_1, \ldots, q_n)$.

We construct $L(p; q_1, \ldots, q_n)$ as follows. We define S^1 as a simplicial set consisting of p non-degenerate 1 and p non-degenerate 0-simplices arranged in a circle. Now we can construct $\rho_i : S^1 \to S^1$ as a morphism of simplicial sets that sends the k^{th} 1-simplex to the $(k + q_i \mod p)^{\text{th}}$ 1-simplex. As we implemented joins of morphisms of simplicial sets, we are able to construct F as the join $\rho_1 \star \rho_2 \star \cdots \star \rho_n$. To obtain $L(p; q_1, \ldots, q_n)$ we take the coequalizer of the identity and F.

5.2. Calculation with groups

For all calculations involving groups we use the [6, GAP] interface that is provided with [17, SageMath]. In particular, we use a GAP function that simplifies group presentations to get more attractive results.

One algorithm we implemented, that is not explicitly given by one of the algorithms described in this paper, is a SageMath function we called all_r_torsions. The function takes as input a finite, connected simplicial set X and, firstly, computes a presentation of its fundamental group G. If GAP can detect that G is finite, then the function asks GAP to compute all irreducible representations of G. For every irreducible representation ϕ of G, we ask GAP to provide the matrix $\phi(e_i)$ for every generator e_i of the computed presentation of G. The function all_r_torsions then returns the Reidemeister torsions of X with respect to the computed representations. The last part of the function is implemented as described in algorithm 8.

5.3. Calculation with matrices

For the computation of the torsion of an acyclic, bounded and based chain complex of finite dimensional $\mathbb{C}$-vector spaces, we implemented algorithm 7 in [18][SageMath]. For the singular value decomposition we used a [7][NumPy] function. We implemented torsion of chain complexes to be computed with double precision floats, so we can not expect more than 15 significant decimal digits precision.
Also, our implementation of algorithm 8 yields a twist complex where the boundary operators are given by matrices with entries double precision complex numbers. The field of double precision complex numbers is refered to by “CDF” in [18][SageMath].

References

[1] Peter I. Booth and Ronald Brown. "Spaces of partial maps, fibred mapping spaces and the compact-open topology". In: *General Topology and its Applications* 8.2 (1978), pp. 181–195. ISSN: 0016-660X. DOI: `https://doi.org/10.1016/0016-660X(78)90049-1`. URL: `https://www.sciencedirect.com/science/article/pii/0016660X78900491`.

[2] TA Chapman. "Topological invariance of Whitehead torsion". In: *American Journal of Mathematics* 96.3 (1974), pp. 488–497.

[3] Marshall M Cohen. *A course in simple-homotopy theory*. Vol. 10. Springer Science & Business Media, 2012.

[4] Greg Friedman. *An elementary illustrated introduction to simplicial sets*. 2021. arXiv: `0809.4221 [math.AT]`.

[5] Peter Gabriel and Michel Zisman. *Calculus of fractions and homotopy theory*. Vol. 35. Springer Science & Business Media, 2012.

[6] *GAP – Groups, Algorithms, and Programming, Version 4.12.2*. The GAP Group. 2023. URL: `%5Curl%7Bhttps://www.gap-system.org%7D`.

[7] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[8] Allen Hatcher. "Algebraic Topology". In: *Hatcher's website* (2002).

[9] John Lee. *Introduction to topological manifolds*. Vol. 202. Springer Science & Business Media, 2010.

[10] Jacob Lurie. *Kerodon*. `https://kerodon.net`. 2023.

[11] John Milnor. "Whitehead torsion". In: *Bulletin of the American Mathematical Society* 72.3 (1966), pp. 358–426.

[12] nLab authors. *adjoint functor*. `https://ncatlab.org/nlab/show/adjoint+functor`. Revision 128. May 2023.

[13] nLab authors. *geometric realization*. `https://ncatlab.org/nlab/show/geometric+realization`. Revision 49. May 2023.

[14] nLab authors. *over category*. `https://ncatlab.org/nlab/show/over+category`. Revision 49. June 2023.

[15] nLab authors. *parameterized homotopy theory*. `https://ncatlab.org/nlab/show/parameterized+homotopy+theory`. Revision 24. June 2023.

[16] nLab authors. *pullback*. `https://ncatlab.org/nlab/show/pullback`. Revision 50. June 2023.

[17] John H. Palmieri. *Sage 10.0 References Manual: Simplicial Sets*. `https://doc.sagemath.org/html/en/reference/topology/sage/topology/simplicial_set.html`.

[18] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5)*. `https://www.sagemath.org`. 2023.

[19] The Stacks project authors. *The Stacks project*. `https://stacks.math.columbia.edu`. 2023.

[20] Vladimir Turaev. *Introduction to Combinatorial Torsions*. Springer Basel AG, 2001.

A. Appendix: Example Calculations and Source Code

The following is an export of the Jupyter notebook available under `https://github.com/fNeugebauer/Computation-of-Combinatorial-Torsion`.

We implemented, among others, the following functions in [18][SageMath]:

1) A function with has input a simplicial set X and a normal subgroup $N \subseteq \pi_1(X)$ with finite index. The function returns a covering projection $p : Z \to X$ such that $p_*\pi_1(Z) = N$ and Z is connected.

2) A function with has input a simplicial set X and a normal subgroup $N \subseteq \pi_1(X)$ with finite index. The function returns a covering projection $p : Z \to X$ such that $p_*\pi_1(Z) = N$, Z is connected and p is given as the pullback of a universal covering projection $\mathrm{sk}_{\dim(X)}(EG) \to \mathrm{sk}_{\dim(X)}(BG)$, where $G := \pi_1(X)/N$. The function returns the pullback square.

3) A function that constructs the join of two simplicial sets, as well as, a function that constructs the join of morphisms of simplicial sets.

4) A function that constructs the lens space $L(p; q_1, \dots, q_n)$ for given $p \in \mathbb{N}_{>0}$ and $q_1, \dots, q_n \in \mathbb{Z}$.

5) A function that computes the combinatorial torsion of a bounded, based chain complex of finite-dimensional $\mathbb{C}$-vector spaces.

6) A function that computes the Reidemeister torsion of a given simplicial set X for a given finite-dimensional complex representation $\phi : \pi_1(X) \to \mathbb{C}^n$ of the fundamental group of X.

7) A function that has input a simplicial set X with finite fundamental group G and returns the Reidemeister torsions of X with respect to all complex irreducible representations of G.

Moreover, we computed, among others, the following examples:

1) Some finite index covering spaces of the torus and the surface of genus 2.

2) The Reidemeister torsions of the following lens spaces:
$$L(5; 1, 1),\ L(4; 1, 3, 1),\ L(5; 2, 3),\ L(7; 1, 1),\ L(7; 1, 2),\ \mathbb{R}P^3.$$
We compared these numbers to the theoretical results in [20, thm. 10.6].

3) The Reidemeister torsions of the Poincaré homology sphere.

4) The Reidemeister torsions of the following products:
$$S_2 \times \mathbb{R}P^3,\ L(3; 1, 1) \times \mathbb{R}P^2,\ L(3; 1, 1) \times S^2,\ L(3; 1, 2) \times \mathbb{R}P^2,$$
$$L(3; 1, 2) \times S^2,\ L(3; 1, 2) \times L(3; 1, 1),\ L(3; 1, 2) \times \mathbb{C}P^2,\ \mathbb{R}P^3 \times \mathbb{C}P^2.$$
These examples all satisfied the formula $\tau(X \times Y, \phi \otimes \psi) = \tau(X, \phi)^{\chi(Y)}\tau(Y, \psi)^{\chi(X)}$ for ϕ, ψ representations of $\pi_1(X)$ and $\pi_1(X)$, respectively.

Contents

[1]:
```
from sage.topology.simplicial_set import AbstractSimplex, SimplicialSet
from sage.topology.simplicial_set_morphism import SimplicialSetMorphism
from itertools import product
from math import prod
import numpy
```

For the later use we implement a function that returns a group presentation of the fundamental group of a connected simplicial set X together with a list $gens = [e_0, \dots, e_m]$. The e_i are non-degenerate 1-simplices in X. Moreover, $X_1^{\mathrm{nd}} \setminus \{e_1, \dots, e_m\}$ is the collection of edges of a spanning tree in the 1-skeleton of X. The computed group presentation will have generators $e_1, \dots, e_m$.

Such an algorithm already exists in SageMath and we just modify it slightly.

[2]:
```
# this is a modification of code under the following Copyright
#*****************************************************************************
#  Copyright (C) 2015 John H. Palmieri <palmieri at math.washington.edu>
#
#  Distributed under the terms of the GNU General Public License (GPL)
#                  http://www.gnu.org/licenses/
#******************************************************************************
```

```
def fundamental_group_w_gens(X):
    if len(X.f_vector())<2:
        return FreeGroup(0)/[], []
    if not X.is_connected():
        raise ValueError("the given simplicial set is not connected")
    graph = X.graph() #graph with edges the non-degenerate 1-cells
    edges = [e[2] for e in graph.edges()]
    spanning_tree = [e[2] for e in graph.min_spanning_tree()]
    gens = [e for e in edges if e not in spanning_tree]
    if gens == []:
        return FreeGroup(0)/[], []
    FG = FreeGroup(len(gens), 'e')
    rels = []
    #adding the reltations given by the 2-cells
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in spanning_tree:
                z[i]=FG.one()
            else:
                z[i]=FG.gen(gens.index(e))
        rels.append(z[0]*z[1].inverse()*z[2])
    return FG.quotient(rels), gens
```

1 Normal Covering Spaces

1.1 Chain complex of normal covering space

The following algorithm implements part 2 of algorithm 3 from the enclosed bachelor thesis. The function is not meant to be directly called but will be used by later functions.

[3]:
```
def complexOfCover(X, G, gens, Simplification = False):

    if not X.is_finite():
        raise ValueError("the given simplicial set might not be finite")
    f_vec = X.f_vector()
    if len(f_vec)<2:
        return X.chain_complex()

    if Simplification:
        h = G.simplification_isomorphism()
        G_simp = h.codomain()
        R = G_simp.algebra(ZZ)
    else:
        R = G.algebra(ZZ)

    deckTrnsf = {i: [] for i in range(1, len(f_vec))}
    bdryComplx = {}
    for i in range(1, len(f_vec)):
        bdryComplx[i] = matrix(R, f_vec[i-1], f_vec[i])

    for j in range(f_vec[1]):
        e = X.n_cells(1)[j]
        ind_d0 = X.n_cells(0).index(X.face(e,0))
        ind_d1 = X.n_cells(0).index(X.face(e,1))
        bdryComplx[1][ind_d1,j] -= 1
        if e in gens:
            if Simplification:
                e_titz = h(G.gen(gens.index(e))).Tietze()
                e_simp = G_simp.one()
                for m in range(len(e_titz)):
                    e_simp = e_simp*G_simp.gen(abs(e_titz[m])-1)^(sgn(e_titz[m]))
                bdryComplx[1][ind_d0,j] += e_simp*R.one()
                deckTrnsf[1].append(e_simp*R.one())
            else:
                bdryComplx[1][ind_d0,j] += R.algebra_generators()[gens.index(e)]
                deckTrnsf[1].append(R.algebra_generators()[gens.index(e)])
        else:
            bdryComplx[1][ind_d0,j] += 1
            deckTrnsf[1].append(1)

    for n in range(2, len(f_vec)):
        for j in range(f_vec[n]):
            f = X.n_cells(n)[j]
            for m in range(1, n+1):
                if X.face(f,m).is_nondegenerate():
                    bdryComplx[n][X.n_cells(n-1).index(X.face(f,m)),j] += (-1)^m
            dn = X.face(f,n)
            if dn.is_degenerate() and 0 in dn.degeneracies():
                if X.face(f,0).is_nondegenerate():
                    bdryComplx[n][X.n_cells(n-1).index(X.face(f,0)),j] += 1
                deckTrnsf[n].append(R.one())
            else:
                dim_und = dn.nondegenerate().dimension()
                ind_und = X.n_cells(dim_und).index(dn.nondegenerate())
                if X.face(f,0).is_nondegenerate():
                    i = X.n_cells(n-1).index(X.face(f,0))
                    bdryComplx[n][i,j] += deckTrnsf[dim_und][ind_und]
                deckTrnsf[n].append(deckTrnsf[dim_und][ind_und])

    return bdryComplx
```

1.1.1 Variant 1: User has to compute spanning tree

The next function implements algorithm 3 of the enclosed bachelor thesis in the unmodified version.

This version of the algorithm requires the user to give a finite simplicial set X together with edges $e_1, \dots, e_m \in X_1^{\mathrm{nd}}$ contained in a list *gens* $= [e_1, \dots, e_m]$ such that $X_1^{\mathrm{nd}} \setminus \{e_1, \dots, e_m\}$ form the edges in a spanning tree of the 1-skeleton of X. Whether this is satisfied, is checked by the following implementation. Further, the user needs to provide a list N of elements in the free group $F(\{e_1, \dots, e_m\})$.

The algorithm computes the normalized chain complex C of a covering space Z of X, such that Z has fundamental group equal to the normal subgroup of $\pi_1(X)$ generated by the elements in N. We will denote the quotient of $\pi_1(X)$ by this subgroup by G. C is a chain complex of free $\mathbb{Z}[G]$-modules, where lifts of the nondegenerate k-simplices of X form a $\mathbb{Z}[G]$-basis of C_k.

Firstly, the function computes a presentation of G such that the $e_1, \dots, e_m$ are the generators.

Then, the function returns C by giving the matrices of the boundary operators with respect to a $\mathbb{Z}[G]$-basis of every C_k, where the basis of C_k consists of lifts of nondegenerate k-simplices of X.

If the function is called with the boolean simplification set to True, then GAP is asked to simplify the computed presentation of G.

[4]:
```
def complexOfNormalCoveringTree(X, gens, N, Simplification=False):

    #X the simplicial set,
    #gens the list of edges not in spanning tree,
    #N relations in the free group F(gens)

    if X.dimension()<1:
        return X.chain_complex()
    if not X.is_connected():
        raise ValueError("given simplicial set not connected")

    #next we check wether sk_1(X)-gens is a spanning tree in sk_1(X)
    treeSimplices = {}
    for e in [item for item in X.n_cells(1) if item not in gens]:
        treeSimplices[e]=(X.face(e,0), X.face(e,1))
    for v in X.n_cells(0):
        treeSimplices[v]=None
    tree = SimplicialSet(treeSimplices)
    if not tree.is_connected():
        raise ValueError("sk_1(X)-gens not spanning subgraph of sk_1(X)")
    if not tree.is_acyclic():
        raise ValueError("sk_1(X)-gens is not acyclic")
    if N == []:
        FG = FreeGroup(len(gens), 'e')
    else:
        if not len(N[0].parent().generators())==len(gens):
            raise ValueError("N not subset of free group on gens")
        FG = N[0].parent()
        for i in range(len(N)):
            if not N[i].parent()==FG:
                raise ValueError("N not subset of one group")

    #now we compute a presentation of pi_1(X)/N
    rels = []
    #adding the relations given by the 2-cells of X
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in gens:
                z[i]=FG.gen(gens.index(e))
            else:
                z[i]=FG.one()
        rels.append(z[0]*z[1].inverse()*z[2])
    G = FG.quotient(rels+N)

    return complexOfCover(X, G, gens, Simplification)
```

Small Examples: We may compute the chain complex of a double cover of the circle, where we define the circle S^1 to be the simplicial set which has only two nondegenerate simplices, one of them being a one-simplex.

```
[5]: s1=simplicial_sets.Sphere(1)
     FG=FreeGroup(1);
     gens = s1.n_cells(1)
     N = [FG.gen(0)^2]
     C_S1_double = complexOfNormalCoveringTree(s1, gens, N)
     C_S1_double, C_S1_double[1].base_ring()
```

```
[5]: ({1: [-1 + x]},
      Algebra of Finitely presented group < x | x^2 > over Integer Ring)
```

Now we compute the chain complex of double cover of the torus.

We use a triangulation of the torus as Δ-complex with two 2-simplices.

```
[6]: t= simplicial_sets.Torus()
     FG=FreeGroup(len(t.n_cells(1)), 'a')
     gens = t.n_cells(1)
     N = [FG.gen(0)^2, FG.gen(1)]
     C_double_torus = complexOfNormalCoveringTree(t, gens, N, True)
     C_double_torus[1].base_ring()
```

[6]:
```
Algebra of Finitely presented group < a0 | a0^2 > over Integer Ring
```

We print the computed result “C_double_torus” in latex:

$$\left\{1 : \begin{pmatrix} -1+a_0 & 0 & -1+a_0 \end{pmatrix}, \quad 2 : \begin{pmatrix} 1 & 1 \\ a_0 & 1 \\ -1 & -1 \end{pmatrix}\right\}$$

1.1.2 Variant 2: User gives the normal subgroup by loops

Next we implement algorithm 3 from the enclosed bachelor thesis in the version modified by algorithm 4.

So, the function has input X, a finite connected simplicial set, and N, where N is a list of lists, $N = [n_0, \ldots, n_s]$.

Each n_i is a list of tuples $(\sigma_{i_1}, k_{i_1}), \ldots, (\sigma_{i_{r_i}}, k_{i_{r_i}})$ for each σ_{i_j} a nondegenerate 1-simplex of X and $k_{i_j} \in \mathbb{Z}$ such that $\sigma_{i_1}^{k_{i_1}} * \cdots * \sigma_{i_{r_i}}^{k_{i_{r_i}}}$ is forming a loop in the 1-skeleton of X.

The condition that n_i defines a loop says that

$$\text{for all } j = 1, \ldots, r_i \quad \begin{cases} d_0(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} > 0 \\ d_0(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} > 0 \text{ and } k_{i_{j+1 \bmod r_i}} < 0 \\ d_1(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} < 0 \text{ and } k_{i_{j+1 \bmod r_i}} > 0 \\ d_1(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} < 0. \end{cases}$$

and $d_0(\sigma_{i_j}) = d_1(\sigma_{i_j})$ if $|k_{i_j}| > 1$.

The algorithm computes the normalized chain complex C of a covering space Z of X, such that Z has fundamental group equal to the normal subgroup of $\pi_1(X)$ generated by the elements in N. We will denote the quotient of $\pi_1(X)$ by this subgroup by G. C is a chain complex of free $\mathbb{Z}[G]$-modules, where lifts of the nondegenerate k-simplices of X form a $\mathbb{Z}[G]$ basis of C_k.

The function computes a presentation of G and returns C by giving the matrices of the boundary operators with respect to a $\mathbb{Z}[G]$-basis of every C_k, where the basis of C_k consists of lifts of nondegenerate k-simplices of X.

If the function is called with the boolean Simplification set to True, then GAP is asked to simplify the computed presentation of G.

If the function is called with the boolean check set to True, then it is check whether N is of the above form.

[7]:
```
def complexOfNormalCoveringLoops(X, N, Simplification=False, check=True):

    #X a finite connected simplicial set,
    #N loops in the 1-skeleton of X given in the above form

    if X.dimension()<1:
        return X.chain_complex()
    if not X.is_connected():
        raise ValueError("given simplicial set not connected")

    #We  check whether N is of the required form:
    if check:
        for n in N:
            for j in range(len(n)):
                s,k = n[j]
                s1, k1 = n[(j+1)%len(n)]
                if abs(k)>1 and X.face(s, 0)!=X.face(s, 1):
                    raise ValueError("N doens't consist of loops")
                if k>0 and k1>0 and X.face(s, 0) != X.face(s1, 1):
                    raise ValueError("N doens't consist of loops")
                if k>0 and k1<0 and X.face(s, 0) != X.face(s1, 0):
                    raise ValueError("N doens't consist of loops")
                if k<0 and k1>0 and X.face(s, 1)!= X.face(s1, 1):
                    raise ValueError("N doens't consist of loops")
                if k<0 and k1<0 and X.face(s, 1)!= X.face(s1, 0):
                    raise ValueError("N doens't consist of loops")

    # let us compute a presentation of the fundamental group G of X
    graph = X.graph() #graph with edges the non-degenerate 1-cells
    edges = [e[2] for e in graph.edges()]
    spanning_tree = [e[2] for e in graph.min_spanning_tree()]
    gens = [e for e in edges if e not in spanning_tree]
    FG = FreeGroup(len(gens), 'e')
    rels = []
    #adding the relations given by the 2-cells
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in spanning_tree:
                z[i]=FG.one()
            else:
                z[i]=FG.gen(gens.index(e))
        rels.append(z[0]*z[1].inverse()*z[2])

    # presentation of normal subgroup of pi_1(X) generated by N
    N_pr = []
    for n in N:
        n_pr=FG.one()
        for e,k in n:
            if e in gens:
                n_pr=n_pr*FG.gen(gens.index(e))^k
        N_pr.append(n_pr)
    G = FG.quotient(rels+N_pr)

    return complexOfCover(X, G, gens, Simplification)
```

Small Examples We use a triangulation of the torus as Δ-complex with two 2-simplices.

```
[8]: torus = simplicial_sets.Torus()
```

```
[9]: N=[[(torus.n_cells(1)[2], 2)]]
     N.append([(torus.n_cells(1)[2], -1)])
     N.append([(torus.n_cells(1)[0], -3), (torus.n_cells(1)[1], 1)])
```

```
[10]: C_torus_N = complexOfNormalCoveringLoops(torus, N, Simplification=True)
      C_torus_N[1].base_ring()
```

```
[10]: Algebra of Finitely presented group < e0 | e0^4 > over Integer Ring
```

We have computed the chain complex of a 4-sheeted covering with deck-transformation group $\mathbb{Z}/4\mathbb{Z}$.

We print “C_torus_N” in latex:

$$\left\{1 : \left(\begin{array}{ccc} -1 + e_0 & -1 + e_0^{-1} & 0 \end{array}\right), \quad 2 : \left(\begin{array}{cc} 1 & e_0^{-1} \\ e_0 & 1 \\ -1 & -1 \end{array}\right)\right\}$$

1.1.3 Variant 3: Chain complex of the universal cover

Computing the normalized chain complex of the universal cover of X as free $\mathbb{Z}[\pi_1(X)]$-module chain complex with basis given by lifts of nondegenerate simplices of X corresponds to computing a normal covering space using the function “complexOfNormalCoveringLoops” and setting $N = 0$ as input.

```
[11]: def complexOfUniversalCover(X, simplify_group=False):
          return complexOfNormalCoveringLoops(X, [], simplify_group, check=False)
```

Examples We use a triangulation of the torus as Δ-complex with two 2-simplices.

```
[12]: torus = simplicial_sets.Torus()
      C_torus = complexOfUniversalCover(torus, simplify_group=True)
      C_torus[1].base_ring()
```

```
[12]: Algebra of Finitely presented group < e0, e1 | e0*e1^-1*e0^-1*e1 > over Integer
      Ring
```

We print “C_torus” by using the function latex(C_torus):

$$\left\{ 1: \begin{pmatrix} -1+e_0 & -1+e_1 & -1+e_0\cdot e_1 \end{pmatrix}, \quad 2: \begin{pmatrix} 1 & e_1 \\ e_0 & 1 \\ -1 & -1 \end{pmatrix} \right\}$$

Next we use a triangulation of the Klein Bottle as Δ-complex with two 2-simplices.

```
[13]: kleinBottle = simplicial_sets.KleinBottle();
      C_kleinBottle= complexOfUniversalCover(kleinBottle, True)
      C_kleinBottle[1].base_ring()
```

```
[13]: Algebra of Finitely presented group < e0, e1 | e0*e1^-1*e0*e1 > over Integer
      Ring
```

Printing the computed chain complex of the universal cover of the Klein bottle in LaTeX gives:

$$\left\{ 1: \begin{pmatrix} -1+e_0 & -1+e_1 & -1+e_0\cdot e_1 \end{pmatrix}, \quad 2: \begin{pmatrix} 1 & e_0\cdot e_1 \\ e_0 & -1 \\ -1 & 1 \end{pmatrix} \right\}$$

Let us try a random simplicial complex of dimension 4 with 11 vertices.

```
[14]: RdCom =SimplicialSet(simplicial_complexes.RandomComplex(11,4))
      RdCom.is_connected()
```

```
[14]: True
```

```
[15]: complexOfUniversalCover(RdCom, simplify_group=True)
```

```
[15]: {1: 11 x 55 dense matrix over Algebra of Finitely presented group <  |  > over
      Integer Ring,
       2: 55 x 165 dense matrix over Algebra of Finitely presented group <  |  > over
      Integer Ring,
       3: 165 x 330 dense matrix over Algebra of Finitely presented group <  |  > over
      Integer Ring,
       4: 330 x 226 dense matrix over Algebra of Finitely presented group <  |  > over
      Integer Ring}
```

RdCom is simply-connected and so we just computed its normalized chain complex

Next we try a triangulation of the Poincaré homology 3-sphere triangulated as simplicial complex.

```
[16]: P_S3 = SimplicialSet(simplicial_complexes.PoincareHomologyThreeSphere())
```

```
[17]: complexOfUniversalCover(P_S3, simplify_group=True)
```

```
[17]: {1: 16 x 106 dense matrix over Algebra of Finitely presented group < e2, e5 |
      e2*e5^-1*e2^-1*e5^-1*e2*e5, e5^-1*e2*e5*e2^-1*e5*e2*e5^-1 > over Integer Ring,
       2: 106 x 180 dense matrix over Algebra of Finitely presented group < e2, e5 |
e2*e5^-1*e2^-1*e5^-1*e2*e5, e5^-1*e2*e5*e2^-1*e5*e2*e5^-1 > over Integer Ring,
 3: 180 x 90 dense matrix over Algebra of Finitely presented group < e2, e5 |
e2*e5^-1*e2^-1*e5^-1*e2*e5, e5^-1*e2*e5*e2^-1*e5*e2*e5^-1 > over Integer Ring}
```

1.2 Construction of normal covering space

The following algorithm is an implementation of part 2 of algorithm 2 from the enclosed bachelor thesis. The following function is not meant to be directly called but will be used in later functions.

[21]:
```
def ConstructionCoveringSpace(X, G, gens):
    if not X.is_finite():
        raise ValueError("given simplicial set might not be finite")
    f_vec = X.f_vector()
    if len(f_vec)<2:
        return X
    G_list = G.list()
    card = len(G_list)

    simp = {n: dict() for n in range(len(f_vec))}
    deckTrnsf = {n: [] for n in range(1, len(f_vec))}
    for n in range(len(f_vec)):
        for s in X.n_cells(n):
            simp[n][s]=[AbstractSimplex(n, name = str(G_list[i])+str(s)) for i␣
↪in range(card)]
    sSet = dict() # the simplicial covering space
    f =dict() #the simplicial covering map
    for v in X.n_cells(0):
        for i in range(card):
            sSet[simp[0][v][i]]= None
            f[simp[0][v][i]] = v
    for e in X.n_cells(1):
        d0 = X.face(e,0); d1 = X.face(e,1)
        if e in gens:
            for i in range(card):
                ie = G_list.index(G_list[i]*G.gen(gens.index(e)))
                sSet[simp[1][e][i]]=( simp[0][d0][ie], simp[0][d1][i])
                f[simp[1][e][i]]= e
            deckTrnsf[1].append(G.gen(gens.index(e)))
        else:
            for i in range(card):
                sSet[simp[1][e][i]]=(simp[0][d0][i], simp[0][d1][i])
                f[simp[1][e][i]]= e
            deckTrnsf[1].append(1)
    for n in range(2, len(f_vec)):
        for s in X.n_cells(n):
            faces = [X.face(s,i).nondegenerate() for i in range(0, n+1)]
            dimns = [faces[i].dimension() for i in range(0,n+1)]
            degens = [X.face(s,i).degeneracies() for i in range(0, n+1)]
            g0 = G.one()
            if degens[n]==[] or min(degens[n])>0:
                g0 = deckTrnsf[dimns[n]][X.n_cells(dimns[n]).index(faces[n])]
            deckTrnsf[n].append(g0)
            for i in range(card):
                ig0 = G_list.index(G_list[i]*g0)
                d0_i_s = simp[dimns[0]][faces[0]][ig0]
                d0_i_s = d0_i_s.apply_degeneracies(*degens[0])
                faces_i_s = [d0_i_s]
                for j in range(1,n+1):
                    dj_i_s = simp[dimns[j]][faces[j]][i]
                    dj_i_s = dj_i_s.apply_degeneracies(*degens[j])
                    faces_i_s.append(dj_i_s)
                sSet[simp[n][s][i]]=tuple(faces_i_s)
                f[simp[n][s][i]]= s
    Z = SimplicialSet(sSet)
    f = SimplicialSetMorphism(f, Z, X)
    return f
```

A small test for the universal cover of simplicial set that is not a Δ-complex.

[19]:
```
S2 = simplicial_sets.Sphere(2)
RP2 = simplicial_sets.RealProjectiveSpace(2)
S2xRP2 = S2.product(RP2)
G, gens = fundamental_group_w_gens(S2xRP2)
```

The universal cover of $S^2 \times \mathbb{R}P^2$ is $S^2 \times S^2$. By the Künneth theorem it has the following reduced homology:

[22]:
```
ConstructionCoveringSpace(S2xRP2, G,gens).domain().homology()
```

[22]:
```
{0: 0, 1: 0, 2: Z x Z, 3: 0, 4: Z}
```

1.2.1 Variant 1: User has to compute spanning tree

The next function implements algorithm 2 in the unmodified version from the bachelor thesis.

This version of the algorithm requires the user to give a finite simplicial set X together with edges $e_1, \ldots, e_m \in X_1^{\text{nd}}$ contained in a list $gens = [e_1, \ldots, e_m]$ such that $X_1^{\text{nd}} \setminus \{e_1, \ldots, e_m\}$ form the edges of a spanning tree in the 1-skeleton of X. Whether this is satisfied, is checked by the following implementation.

Further, the function has as input a list N of elements in the free group $F(\{e_1, \ldots, e_m\})$, such that the quotient of $F(\{e_1, \ldots, e_m\})$ by N union relations in $\pi_1(X)$ is a finite group G.

The algorithm returns a morphism of simplicial sets $f : Z \to X$ such that f is a covering projection and $f_*(\pi_1(Z))$ is the normal subgroup of $\pi_1(X)$ generated by the elements in N. Further, Z is a finite, connected simplicial set.

[23]:
```
def NormalCoveringSpaceTree(X, gens, N):

    #X the simplicial set,
    #gens the list of edges not in spanning tree,
    #N relations in the free group F(gens)

    if X.dimension()<1:
        return X
    if not X.is_connected():
        raise ValueError("given simplicial set not connected")

    #next we check wether sk_1(X)-gens is a spanning tree in sk_1(X)
    treeSimplices = {}
    for e in [item for item in X.n_cells(1) if item not in gens]:
        treeSimplices[e]=(X.face(e,0), X.face(e,1))
    for v in X.n_cells(0):
        treeSimplices[v]=None
    tree = SimplicialSet(treeSimplices)
    if not tree.is_connected():
        raise ValueError("sk_1(X)-gens not spanning subgraph of sk_1(X)")
    if not tree.is_acyclic():
        raise ValueError("sk_1(X)-gens is not acyclic")
    if N == []:
        FG = FreeGroup(len(gens), 'e')
    else:
        if not len(N[0].parent().generators())==len(gens):
            raise ValueError("N not subset of free group on gens")
        FG = N[0].parent()
        for i in range(len(N)):
            if not N[i].parent()==FG:
                raise ValueError("N not subset of one group")

    #now we compute a presentation of pi_1(X)/N
    rels = []
    #adding the relations given by the 2-cells of X
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in gens:
                z[i]=FG.gen(gens.index(e))
            else:
                z[i]=FG.one()
        rels.append(z[0]*z[1].inverse()*z[2])
    G = FG.quotient(rels+N)
    if not G.is_finite():
        raise ValueError("The quotient pi_1(X)/N is not finite")

    return ConstructionCoveringSpace(X, G, gens)
```

Some examples:

A triangulation of the torus $\mathbb{T}$ as Δ-complex with two 2-simplices Let us compute a covering space projection $f : Z \to \mathbb{T}$ such that the quotient $\pi_1(\mathbb{T})/\pi_1(Z)$ equals $\mathbb{Z}/5 \oplus \mathbb{Z}/3$.

```
[24]: t=simplicial_sets.Torus()
      FG=FreeGroup(len(t.n_cells(1)), 'a')
      gens = t.n_cells(1)
      N = [FG.gen(0)^5, FG.gen(2)^3]
      f = NormalCoveringSpaceTree(t, gens, N)
      f.domain(), f.codomain()
```

```
[24]: (Simplicial set with 90 non-degenerate simplices, Torus)
```

Let us check which homomorphism the computed covering projection f induces on homology.

```
[25]: f.induced_homology_morphism().to_matrix()
```

```
[25]: [ 1| 0  0| 0]
      [--+-----+--]
      [ 0|-5  0| 0]
      [ 0| 2  3| 0]
      [--+-----+--]
      [ 0| 0  0|15]
```

Computing Smith-Normal form of the middle matrix, corresponding to $H_1(f)$ tells us that we computed the correct covering space. From the matrix corresponding to $H_2(f)$, we see that the covering projection has degree 15, i.e. f is 15-sheeted.

We repeat this example with different $N's$.

```
[26]: N=[FG.gen(1)^3, FG.gen(0)^2]
      t_cover = NormalCoveringSpaceTree(t, gens, N)
      t_cover.induced_homology_morphism().to_matrix()
```

```
[26]: [1|0 0|0]
      [-+---+-]
      [0|3 1|0]
      [0|0 2|0]
      [-+---+-]
      [0|0 0|6]
```

```
[28]: N=[ FG.gen(1)^2, FG.gen(0)^3]
      t_cover = NormalCoveringSpaceTree(t, t.n_cells(1), N)
      t_cover.induced_homology_morphism().to_matrix()
```

```
[28]: [ 1| 0  0| 0]
      [--+-----+--]
      [ 0| 2 -1| 0]
      [ 0| 0  3| 0]
      [--+-----+--]
      [ 0| 0  0| 6]
```

A non-abelian example: Let us compute the universal cover of the 3-skeleton of the classifying space of the symmetric group on 3-letters.

```
[29]: S3=SymmetricGroup(3); BS3=simplicial_sets.ClassifyingSpace(S3)
```

```
[30]: BS3_k3 = BS3.n_skeleton(3); BS3_k3
```

```
[30]: Simplicial set with 156 non-degenerate simplices
```

```
[32]: gens = BS3_k3.n_cells(1)
      univ_cov_BS3_k3 = NormalCoveringSpaceTree(BS3_k3, gens,[])
```

```
[33]: ES3_sk_3 = univ_cov_BS3_k3.domain(); ES3_sk_3.homology()
```

```
[33]: {0: 0, 1: 0, 2: 0, 3: Z^625}
```

1.2.2 Variant 2: User gives the normal subgroup by loops

Next we implement algorithm 2 from the enclosed bachelor thesis in the version modified by algorithm 4.

So, the function has input X, a finite connected simplicial set, and N, where N is a list of lists, $N = [n_0, \dots, n_s]$.

Each n_i is a list of tuples $(\sigma_{i_1}, k_{i_1}), \dots, (\sigma_{i_{r_i}}, k_{i_{r_i}})$ for each σ_{i_j} a nondegenerate 1-simplex of X and $k_{i_j} \in \mathbb{Z}$ such that $\sigma_{i_1}^{k_{i_1}} * \cdots * \sigma_{i_{r_i}}^{k_{i_{r_i}}}$ is forming a loop in the 1-skeleton of X.

The condition that n_i defines a loop says that

$$\text{for all } j = 1, \dots, r_i \qquad \begin{cases} d_0(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} > 0 \\ d_0(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} > 0 \text{ and } k_{i_{j+1 \bmod r_i}} < 0 \\ d_1(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} < 0 \text{ and } k_{i_{j+1 \bmod r_i}} > 0 \\ d_1(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} < 0. \end{cases}$$

and $d_0(\sigma_{i_j}) = d_1(\sigma_{i_j})$ if $|k_{i_j}| > 1$.

The function returns a simplicial covering space projection $f : Z \to X$ such that $f_*(\pi_1(Z))$ equals the normal subgroup of $\pi_1(X)$ generated by the loops in N. Further, Z is a finite, connected simplicial set.

If the function is called with the boolean check set to True, then it is check whether N is of the above form.

[34]:
```
def NormalCoveringSpaceLoops(X, N, check=True):

    #X a finite connected simplicial set,
    #N loops in the 1-skeleton of X given in the above form

    if X.dimension()<1:
        return X
    if not X.is_connected():
        raise ValueError("given simplicial set not connected")

    #We  check whether N is of the required form:
    if check:
        for n in N:
            for j in range(len(n)):
                s,k = n[j]
                s1, k1 = n[(j+1)%len(n)]
                if abs(k)>1 and X.face(s, 0)!=X.face(s, 1):
                    raise ValueError("N doens't consist of loops")
                if k>0 and k1>0 and X.face(s, 0) != X.face(s1, 1):
                    raise ValueError("N doens't consist of loops")
                if k>0 and k1<0 and X.face(s, 0) != X.face(s1, 0):
                    raise ValueError("N doens't consist of loops")
                if k<0 and k1>0 and X.face(s, 1)!= X.face(s1, 1):
                    raise ValueError("N doens't consist of loops")
                if k<0 and k1<0 and X.face(s, 1)!= X.face(s1, 0):
                    raise ValueError("N doens't consist of loops")

    # let us compute a presentation of the fundamental group G of X
    graph = X.graph() #graph with edges the non-degenerate 1-cells
    edges = [e[2] for e in graph.edges()]
    spanning_tree = [e[2] for e in graph.min_spanning_tree()]
    gens = [e for e in edges if e not in spanning_tree]
    FG = FreeGroup(len(gens), 'e')
    rels = []
    #adding the relations given by the 2-cells
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in spanning_tree:
                z[i]=FG.one()
            else:
                z[i]=FG.gen(gens.index(e))
        rels.append(z[0]*z[1].inverse()*z[2])

    # presentation of normal subgroup of pi_1(X) generated by N
    N_pr = []
    for n in N:
        n_pr=FG.one()
        for e,k in n:
            if e in gens:
                n_pr=n_pr*FG.gen(gens.index(e))^k
        N_pr.append(n_pr)

    G = FG.quotient(rels+N_pr)

    return ConstructionCoveringSpace(X, G, gens)
```

Some examples:

A triangulation of the torus as Δ-complex with two 2-simplices

```
[35]: torus = simplicial_sets.Torus()
```

```
[36]: N=[[(torus.n_cells(1)[1], 2)], [(torus.n_cells(1)[0], -3)]]
```

```
[37]: f = NormalCoveringSpaceLoops(torus, N, check=True)
```

```
[38]: f.domain(), f.codomain()
```

```
[38]: (Simplicial set with 36 non-degenerate simplices, Torus)
```

We computed a 6-sheeted covering map $f : Z \to \mathbb{T}$ of the torus $\mathbb{T}$ such that $f_*\pi_1(Z)$ is generated by $N = \{\alpha^2, \beta^{-3}\}$, where α,β are loops generating $\pi_1(\mathbb{T})$. Let us compute the induced map on homology:

```
[39]: f.induced_homology_morphism().to_matrix()
```

```
[39]: [ 1| 0  0| 0]
      [--+-----+--]
      [ 0| 2 -1| 0]
      [ 0| 0  3| 0]
      [--+-----+--]
      [ 0| 0  0| 6]
```

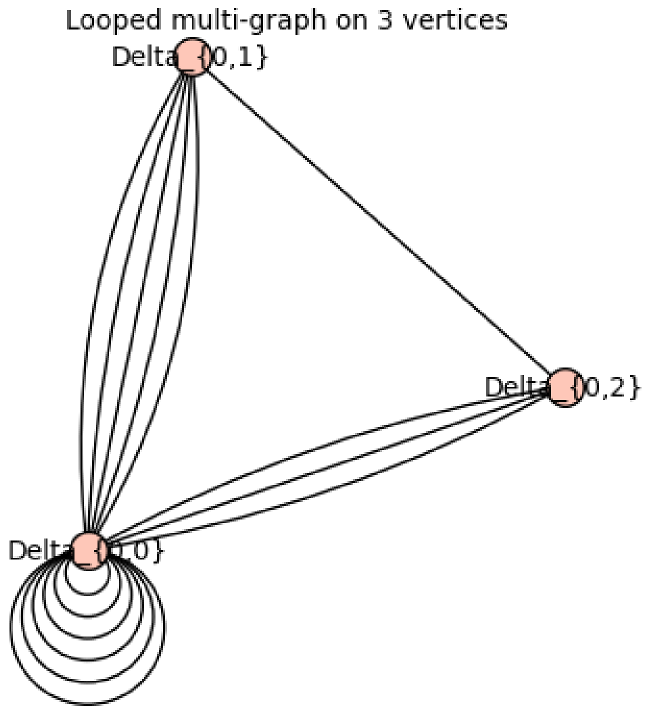
Surface of genus g as connected sum of tori, wich are triangulated as above

```
[40]: Sg = SimplicialSet(delta_complexes.SurfaceOfGenus(2)); Sg
```

```
[40]: Simplicial set with 28 non-degenerate simplices
```

```
[41]: Sg.graph()
```

[41]:



```
[42]: N=[[(Sg.n_cells(1)[i], 1)] for i in [0,1,2]]
      N= N+[[(Sg.n_cells(1)[i], 1),(Sg.n_cells(1)[i], -1)] for i in range(3,6)]
      N=N+[[(Sg.n_cells(1)[i], 2)] for i in [6,7]]
```

We compute a covering space projection $f : Z \to S_g$ such that $f_*(\pi_1(Z)) = N$, where Z is connected.

```
[43]: f = NormalCoveringSpaceLoops(Sg, N, check=True)
```

```
[44]: f.codomain().homology(), f.domain().homology()
```

```
[44]: ({0: 0, 1: Z x Z x Z x Z, 2: Z}, {0: 0, 1: Z^6, 2: Z})
```

```
[45]: f.induced_homology_morphism().to_matrix()
```

```
[45]: [ 1| 0  0  0  0  0  0| 0]
      [--+-----------------+--]
      [ 0| 0  0  2  2  0  0| 0]
      [ 0| 0 -1 -1 -1  1  0| 0]
      [ 0| 0  1  2  1  0  0| 0]
      [ 0| 1  0  1  0  0  1| 0]
      [--+-----------------+--]
      [ 0| 0  0  0  0  0  0| 2]
```

We have computed a double cover of the surface of genus 2 by the surface of genus 3.

1.2.3 Construction of classifying spaces

The following function returns the k-skeleton of the classifying space of a given group. The algorithm is not meant to be used directly as SageMath can already compute classifying spaces. I just implemented this function for later use.

[46]:
```
def k_skeleton_of_classifing_space(G, G_list,card, k):
    #G a finite group, G_list a list of its elements
    #card the cardinality of the group
    #k the dimension of the skeleton

    simp = {}
    #simp will be dictionary with keys given by tuples
    #each tuple will have length leq k entries in range(1,card)
    #the values of simp are the nondegenerate simplices of BG

    simp[0]=AbstractSimplex(0, name = '()' )
    sSet={simp[0]: None }

    # 1-cells
    for s in range(1,card):
        simp[(s,)]=AbstractSimplex(1, name = str(s))
        sSet[simp[(s,)]]= (simp[0],simp[0])

    for n in range(2,k+1):
        for s in product(range(1, card), repeat=n):
            simp[s] = AbstractSimplex(n, name = str(s))
            faces = []
            faces.append(simp[s[1:]])
            for j in range(1,n):
                p = G_list.index(G_list[s[j-1]]*G_list[s[j]])
                if p == 0: #degenerate case
                    if n==2:
                        faces.append(simp[0].apply_degeneracies(0))
                    else:
                        simplex = simp[s[0:(j-1)]+s[(j+1):len(s)]]
                        simplex = simplex.apply_degeneracies(j-1)
                        faces.append(simplex )
                else:
                        faces.append(simp[s[0:(j-1)]+(p,)+s[(j+1):len(s)]] )
            faces.append(simp[s[0:(len(s)-1)]])
            sSet[simp[s]]=tuple(faces)
    BG = SimplicialSet(sSet, name = 'BG', base_point = simp[0])
    return BG
```

Very small example We compute the 2-skeleton of the classifying space of the symmetric group on 4 letters.

```
[47]: H=SymmetricGroup(4)
```

```
[48]: BH_2 = k_skeleton_of_classifing_space(H, H.list(), H.cardinality(), 2)
```

```
[49]: BH_2.fundamental_group().is_isomorphic(SymmetricGroup(4))
```

```
#I  Forcing finiteness test
```

```
[49]: True
```

1.2.4 Classifying map of normal covering space

This implements part 2 of algorithm 1 from the enclosed bachelor thesis. The function is not meant to be directely called by the user, but we will use the funtion in later algorithms.

```
[60]: def construction_f_bar(X, gens, G):
          #X the simplicial set, gens list of edges not in spanning tree,
          #G a presentation of pi_1(X)/N with generators given by gens
          if not X.is_finite():
              raise ValueError("given simplicial set might not be finite")

          f_vec = X.f_vector()

          if not G.is_finite():
              raise ValueError("the quotient pi_1(X)/N is not finite")

          G_list = G.list()
          card = len(G_list)

          #construction of the dim X skeleton of BG:
          simp = {}
          #simp will be dictionary with keys given by tuples
          #each tuple will have length leq k entries in range(1,card)
          #the values of simp are the nondegenerate simplices of BG

          simp[0]=AbstractSimplex(0, name = '()' )
          sSet={simp[0]: None }
          # 1-cells
          for s in range(1,card):
```

```
        simp[(s,)]=AbstractSimplex(1, name = str(s))
        sSet[simp[(s,)]]= (simp[0],simp[0])
    for n in range(2,len(f_vec)):
        for s in product(range(1, card), repeat=n):
            simp[s] = AbstractSimplex(n, name = str(s))
            faces = []
            faces.append(simp[s[1:]])
            for j in range(1,n):
                p = G_list.index(G_list[s[j-1]]*G_list[s[j]])
                if p == 0: #degenerate case
                    if n==2:
                        faces.append(simp[0].apply_degeneracies(0))
                    else:
                        simplex = simp[s[0:(j-1)]+s[(j+1):len(s)]]
                        simplex = simplex.apply_degeneracies(j-1)
                        faces.append(simplex )
                else:
                        faces.append(simp[s[0:(j-1)]+(p,)+s[(j+1):len(s)]] )
            faces.append(simp[s[0:(len(s)-1)]])
            sSet[simp[s]]=tuple(faces)
    BG = SimplicialSet(sSet, name = 'BG', base_point = simp[0])

    # the morphism of simplicial sets X to BG that classifies X
    f = {v: simp[0] for v in X.n_cells(0)}
    for e in X.n_cells(1):
        if e in gens:
            beta_e = G_list.index(G.gen(gens.index(e)))
            if beta_e == 0:
                f[e]=simp[0].apply_degeneracies(0)
            else:
                f[e]=simp[(beta_e,)]
        else:
            f[e]=simp[0].apply_degeneracies(0)
    for n in range(2, len(f_vec)):
        for sigma in X.n_cells(n):
            d0 = X.face(sigma,0)
            dn=X.face(sigma,n)
            degens_n = dn.degeneracies()
            und_f_d0 = f[d0.nondegenerate()].nondegenerate()
            und_f_dn = f[dn.nondegenerate()].nondegenerate()
            degens_f_d0 = f[d0.nondegenerate()].degeneracies()
            degens_f_dn = f[dn.nondegenerate()].degeneracies()
            s0 = [s for s, simplex in simp.items() if simplex == und_f_d0][0]
            #s0 is und_f_d0 as tuple
            sn = [s for s, simplex in simp.items() if simplex == und_f_dn][0]

            dim_d0 = d0.dimension()
            dim_d0_nd = d0.nondegenerate().dimension()

            cond = False
            if und_f_d0.dimension()==0:
                cond = True
            if d0.degeneracies() != [] and d0.degeneracies()[0]>=dim_d0-1:
                cond = True
            if degens_f_d0 != [] and degens_f_d0[0]>=dim_d0_nd-1:
                cond = True

            if und_f_dn.dimension()==0 and cond:
                f[sigma]=simp[0].apply_degeneracies(*degens_f_dn)
                f[sigma]=f[sigma].apply_degeneracies(*degens_n)
                f[sigma]=f[sigma].apply_degeneracies(n-1)
            if und_f_dn.dimension()==0 and not cond:
                f[sigma]=simp[(s0[-1],)].apply_degeneracies(*degens_f_dn)
                f[sigma]=f[sigma].apply_degeneracies(*degens_n)
            if und_f_dn.dimension()!=0 and cond:
                f[sigma]=simp[sn].apply_degeneracies(*degens_f_dn)
                f[sigma]=f[sigma].apply_degeneracies(*degens_n)
                f[sigma]=f[sigma].apply_degeneracies(n-1)
            if und_f_dn.dimension()!=0 and not cond:
                f[sigma]=simp[sn+(s0[-1],)].apply_degeneracies(*degens_f_dn)
                f[sigma]=f[sigma].apply_degeneracies(*degens_n)

    return SimplicialSetMorphism(f, X, BG), BG, simp, G, G_list, card
```

1.2.5 Normal covering space by classifying map Variant 1: User has to compute spanning tree

This function implements algorithm 1 from the enclosed bachelor thesis in the unmodified version but omits the construction of the pullback.

This version of the algorithm requires the user to give a finite simplicial set X together with edges $e_1, \ldots, e_m \in X_1^{\mathrm{nd}}$ contained in a list *gens* $= [e_1, \ldots, e_m]$ such that $X_1^{\mathrm{nd}} \setminus \{e_1, \ldots, e_m\}$ form the edges of a spanning tree in the 1-skeleton of X. Whether this is satisfied, is checked by the following implementation.

Further, the function has as input a list N of elements in the free group $F(\{e_1, \ldots, e_m\})$, such that the quotient of $F(\{e_1, \ldots, e_m\})$ by N union relations in $\pi_1(X)$ is a finite group G.

The function returns a morphism of simplicial sets $f : X \to BG$, where BG is the $\dim(X)$-skeleton of the classifying space of G. Further, $\pi_1(f)$ is surjective. Moreover, the pullback of the canonical map from the $\dim(X)$-skeleton of EG to BG along f yields a covering projection $f : Z \to X$, such that Z is a finite connected simplicial set and $f_*(\pi_1(Z))$ is the normal subgroup generated by the elements of N.

```
[51]: def classifying_map_tree(X, gens, N):
          return classifying_map_normal_covering_tree(X, gens, N)[0]
```

```
def classifying_map_normal_covering_tree(X, gens, N):
    #X the simplicial set,
    #gens the list of edges not in spanning tree,
    #N relations in the free group F(gens)

    if not X.is_connected():
        raise ValueError("given simplicial set not connected")

    #next we check wether sk_1(X)-gens is a spanning tree in sk_1(X)
    treeSimplices = {}
    for e in [item for item in X.n_cells(1) if item not in gens]:
        treeSimplices[e]=(X.face(e,0), X.face(e,1))
    for v in X.n_cells(0):
        treeSimplices[v]=None
    tree = SimplicialSet(treeSimplices)
    if not tree.is_connected():
        raise ValueError("sk_1(X)-gens not spanning subgraph of sk_1(X)")
    if not tree.is_acyclic():
        raise ValueError("sk_1(X)-gens is not acyclic")
    if N == []:
        FG = FreeGroup(len(gens), 'e')
    else:
        if not len(N[0].parent().generators())==len(gens):
            raise ValueError("N not subset of free group on gens")
        FG = N[0].parent()
        for i in range(len(N)):
            if not N[i].parent()==FG:
                raise ValueError("N not subset of one group")

    #now we compute a presentation of pi_1(X)/N
    rels = []
    #adding the relations given by the 2-cells of X
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in gens:
                z[i]=FG.gen(gens.index(e))
            else:
                z[i]=FG.one()
        rels.append(z[0]*z[1].inverse()*z[2])

    G = FG.quotient(rels+N)
    if not G.is_finite():
        raise ValueError("The quotient pi_1(X)/N is not finite")

    return construction_f_bar(X,gens,G)
```

Two small examples: A triangulation of the torus as Δ-complex with two 2-simplices and a map classifying a finite sheeted covering space.

```
[56]: t = simplicial_sets.Torus()
      gens = t.n_cells(1)
      FG=FreeGroup(len(gens), 'a')
      N = [FG.gen(0)^7, FG.gen(1)^8]
      classifying_map_tree(t, gens, N)
```

```
[56]: Simplicial set morphism:
        From: Torus
        To:   BG
        Defn: [(v_0, v_0), (s_0 v_0, sigma_1), (sigma_1, s_0 v_0), (sigma_1, sigma_1),
      (s_0 sigma_1, s_1 sigma_1), (s_1 sigma_1, s_0 sigma_1)] --> [(), 1, 3, 5, (1,
      3), (3, 1)]
```

An example of simplicial sets which are not Δ-complexes.

```
[61]: S2 = simplicial_sets.Sphere(2)
      RP2 = simplicial_sets.RealProjectiveSpace(2)
      S2xRP2 = S2.product(RP2)
      G, gens = fundamental_group_w_gens(S2xRP2)
      classifying_map_tree(S2xRP2, gens,[])
```

```
[61]: Simplicial set morphism:
        From: S^2 x RP^2
        To:   BG
        Defn: [(v_0, 1), (s_0 v_0, f), (s_1 s_0 v_0, f * f), (sigma_2, f * f),
      (sigma_2, s_0 f), (sigma_2, s_1 f), (sigma_2, s_1 s_0 1), (s_0 sigma_2, s_1 f *
      f), (s_0 sigma_2, s_2 f * f), (s_0 sigma_2, s_2 s_1 f), (s_1 sigma_2, s_0 f *
      f), (s_1 sigma_2, s_2 f * f), (s_1 sigma_2, s_2 s_0 f), (s_2 sigma_2, s_0 f *
      f), (s_2 sigma_2, s_1 f * f), (s_2 sigma_2, s_1 s_0 f), (s_1 s_0 sigma_2, s_3
      s_2 f * f), (s_2 s_0 sigma_2, s_3 s_1 f * f), (s_2 s_1 sigma_2, s_3 s_0 f * f),
      (s_3 s_0 sigma_2, s_2 s_1 f * f), (s_3 s_1 sigma_2, s_2 s_0 f * f), (s_3 s_2
      sigma_2, s_1 s_0 f * f)] --> [(), 1, (1, 1), (1, 1), s_0 1, s_1 1, s_1 s_0 (),
      s_1 (1, 1), s_2 (1, 1), s_2 s_1 1, s_0 (1, 1), s_2 (1, 1), s_2 s_0 1, s_0 (1,
      1), s_1 (1, 1), s_1 s_0 1, s_3 s_2 (1, 1), s_3 s_1 (1, 1), s_3 s_0 (1, 1), s_2
      s_1 (1, 1), s_2 s_0 (1, 1), s_1 s_0 (1, 1)]
```

Normal covering projection as pullback from classifying map The following function implements all parts of algorithm 1 from the enclosed bachelor thesis in the unmodified version.

This version of the algorithm requires the user to give a finite simplicial set X together with edges

$e_1, \dots, e_m \in X_1^{\mathrm{nd}}$ contained in a list $gens = [e_1, \dots, e_m]$ such that $X_1^{\mathrm{nd}} \setminus \{e_1, \dots, e_m\}$ form the edges of a spanning tree in the 1-skeleton of X. Whether this is satisfied, is checked by the following implementation.

Further, the function has as input a list N of elements in the free group $F(\{e_1, \dots, e_m\})$, such that the quotient of $F(\{e_1, \dots, e_m\})$ by N union relations in $\pi_1(X)$ is a finite group G.

The function returns the pullback Z of the diagram $X \xrightarrow{f} BG \leftarrow EG$, where BG is the $\dim(X)$-skeleton of the classifying space of G, EG is a universal cover of BG and f is a morphism of simplicial sets constructed in such a way that the following hold: 1) The canonical map $p : Z \to X$ is a covering projection. 2) Z is connected 3) $p_*(\pi_1(Z))$ is the subgroup of $\pi_1(X)$ generated by the elements of N 3) $\pi_1(f)$ is surjective.

```
[62]: def NormalCoveringSpaceFromClassifyingSpaceTree(X, gens, N):

          #X the simplicial set,
          #gens the list of edges not in spanning tree,
          #N relations in the free group F(gens)
          f, BG,simpBG, G, G_list,card=classifying_map_normal_covering_tree(X,gens,N)

          # construction of the dim X skeleton of EG and the covering map q: EG to BG
          simp = {(s,): AbstractSimplex(0, name='('+str(s)+')') for s in range(card)}

          #simp is a dictionary with keys given by tuples
          #each tuples has entries in range(card) and length leq dim X
          #the values of the dictionary simp are the simplices of EG

          q = {simp[(s,)]: simpBG[0] for s in range(0,card)}
          sSet={simp[(s,)]: None for s in range(0, card)}

          for n in range(1,X.dimension()+1):
              for s in product(range(0, card), repeat=(n+1)):
                  if not any([s[i]==s[i+1] for i in range(0, n)]):
                      # s gives non-degenerate of EG
                      simp[s] = AbstractSimplex(n, name = str(s))
                      faces = []
                      for i in range(0, n+1):
                          if i>0 and i<n and s[i-1]==s[i+1]:
                              simplex = simp[s[0:i]+s[(i+2):len(s)]]
                              simplex = simplex.apply_degeneracies(i-1)
                              faces.append(simplex)
                          else:
                              faces.append(simp[s[0:i]+s[(i+1):len(s)]])
                      sSet[simp[s]]=tuple(faces)
                      g_s = [G_list[s[i]].inverse()*G_list[s[i+1]] for i in range(n)]
                      q_s = [G_list.index(g) for g in g_s]
                      q[simp[s]]=simpBG[tuple(q_s)]
          EG = SimplicialSet(sSet, name='EG', base_point=simp[(0,)])
```

```
    q_mor = SimplicialSetMorphism(q, EG, BG)
    return BG.pullback(f, q_mor)
```

Example: A triangulation of the torus as Δ-complex with two 2-simplices

[63]:
```
t = simplicial_sets.Torus()
```

[64]:
```
gens =  t.n_cells(1)
FG=FreeGroup(len(gens))
N = [FG.gen(0)^3, FG.gen(1)^3]
```

[65]:
```
Z= NormalCoveringSpaceFromClassifyingSpaceTree(t, gens, N)
```

We extract the covering projection from the pullback. Then, we check, whether Z has the reduced homology of a torus.

[66]:
```
p=Z.structure_map(0); Z.homology()
```

[66]:
```
{0: 0, 1: Z x Z, 2: Z}
```

The covering projection p induces the following map on homology:

[591]:
```
p_star = p.induced_homology_morphism(); p_star.to_matrix()
```

[591]:
```
[ 1| 0  0| 0]
[--+-----+--]
[ 0| 3 -3| 0]
[ 0| 0  3| 0]
[--+-----+--]
[ 0| 0  0| 9]
```

We see that p has degree 9 and Z is a 9-sheeted covering space of the torus.

Let us do an example with different N:

[67]:
```
N=[FG.gen(1)^4, FG.gen(2)^3]
```

[68]:
```
Z= NormalCoveringSpaceFromClassifyingSpaceTree(t, gens, N)
```

We extract the covering projection from the pullback. Then, we check, whether Z has the reduced homology of a torus.

[69]:
```
p=Z.structure_map(0); Z.homology()
```

[69]:
```
{0: 0, 1: Z x Z, 2: Z}
```

The covering projection p induces the following map on homology:

[70]:
```
p_star = p.induced_homology_morphism(); p_star.to_matrix()
```

```
[70]: [ 1| 0  0| 0]
      [--+-----+--]
      [ 0| 4 -8| 0]
      [ 0|-3  9| 0]
      [--+-----+--]
      [ 0| 0  0|12]
```

By definition of N the subgroup of $\pi_1(\mathbb{T})$ generated by the elements of N is given by $4\mathbb{Z} \oplus 3\mathbb{Z}$. So, the quotient of $\pi_1(\mathbb{T})$ by this subgroup is given by $\mathbb{Z}/(4) \oplus \mathbb{Z}/(3)$, which is isomorphic to $\mathbb{Z}/12$, since 3 and 4 are coprime.

Indeed:

```
[71]: BG=Z.defining_map(0).codomain(); BG.homology()
```

```
[71]: {0: 0, 1: C12, 2: Z^110}
```

Bringing the matrix of $H_1(p)$ computed above in Smith normal form yields:

```
[72]: matrix([[4, -8],[-3,9]]).smith_form()
```

```
[72]: (
      [ 1  0]  [1 1]  [ 1 -1]
      [ 0 12], [3 4], [ 0  1]
      )
```

So, indeed $H_1(\mathbb{T})/p_*(H_1(Z)) \cong \mathbb{Z}/12$.

1.2.6 Normal covering space by classifying map Variant 2: User gives the normal subgroup by loops

This implements algorithm 1 from the enclosed bachelor thesis in the unmodified version but omits the construction of the pullback.

So, the function has input X, a finite connected simplicial set, and N, where N is a list of lists, $N = [n_0, \dots, n_s]$.

Each n_i is a list of tuples $(\sigma_{i_1}, k_{i_1}), \dots, (\sigma_{i_{r_i}}, k_{i_{r_i}})$ for each σ_{i_j} a nondegenerate 1-simplex of X and $k_{i_j} \in \mathbb{Z}$ such that $\sigma_{i_1}^{k_{i_1}} * \cdots * \sigma_{i_{r_i}}^{k_{i_{r_i}}}$ is forming a loop in the 1-skeleton of X.

The condition that n_i defines a loop says that

$$\text{for all } j = 1, \dots, r_i \quad \begin{cases} d_0(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} > 0 \\ d_0(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} > 0 \text{ and } k_{i_{j+1 \bmod r_i}} < 0 \\ d_1(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} < 0 \text{ and } k_{i_{j+1 \bmod r_i}} > 0 \\ d_1(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} < 0. \end{cases}$$

and $d_0(\sigma_{i_j}) = d_1(\sigma_{i_j})$ if $|k_{i_j}| > 1$.

The function returns a morphism of simplicial sets $f : X \to BG$, where BG is the dim(X)-skeleton of the classifying space of G. Further, $\pi_1(f)$ is surjective. Moreover, the pullback of the canonical map from the dim(X)-skeleton of EG to BG along f yields a covering projection $f : Z \to X$, such that Z is a finite connected simplicial set and $f_*(\pi_1(Z))$ is the normal subgroup generated by the elements of N.

If the function is called with the boolean check set to True, then it is check whether N is of the above form.

[73]:
```
def classifying_map_loops(X, N, check=True):
    return classifying_map_normal_covering_loops(X, N, check)[0]

def classifying_map_normal_covering_loops(X, N, check=True):

    #X a finite connected simplicial set,
    #N loops in the 1-skeleton of X given in the above form

    if not X.is_connected():
        raise ValueError("given simplicial set not connected")

    #We  check whether N is of the required form:
    if check:
        for n in N:
            for j in range(len(n)):
                s,k = n[j]
                s1, k1 = n[(j+1)%len(n)]
                if abs(k)>1 and X.face(s, 0)!=X.face(s, 1):
                    raise ValueError("N doens't consist of loops")
                if k>0 and k1>0 and X.face(s, 0) != X.face(s1, 1):
                    raise ValueError("N doens't consist of loops")
                if k>0 and k1<0 and X.face(s, 0) != X.face(s1, 0):
                    raise ValueError("N doens't consist of loops")
                if k<0 and k1>0 and X.face(s, 1)!= X.face(s1, 1):
                    raise ValueError("N doens't consist of loops")
                if k<0 and k1<0 and X.face(s, 1)!= X.face(s1, 0):
                    raise ValueError("N doens't consist of loops")

    #we compute a presentation of the fundamental group G of X
    graph = X.graph() #graph with edges the non-degenerate 1-cells
    edges = [e[2] for e in graph.edges()]
    spanning_tree = [e[2] for e in graph.min_spanning_tree()]
    gens = [e for e in edges if e not in spanning_tree]
    FG = FreeGroup(len(gens), 'e')
    rels = []
    #adding the relations given by the 2-cells
    for f in X.n_cells(2):
        z = dict()
        for i in range(3):
            e = X.face(f,i)
            if e.is_degenerate():
                z[i]=FG.one()
            elif e in spanning_tree:
                z[i]=FG.one()
            else:
                z[i]=FG.gen(gens.index(e))
        rels.append(z[0]*z[1].inverse()*z[2])

    # presentation of normal subgroup of pi_1(X) generated by N
    N_pr = []
    for n in N:
        n_pr=FG.one()
        for e,k in n:
            if e in gens:
                n_pr=n_pr*FG.gen(gens.index(e))^k
        N_pr.append(n_pr)

    G = FG.quotient(rels+N_pr)

    return construction_f_bar(X,gens,G)
```

A example: Triangulation of the torus as Δ-complex with two 2-simplices.

```
[74]: torus = simplicial_sets.Torus()
```

```
[75]: N=[[(torus.n_cells(1)[1], 2)], [(torus.n_cells(1)[0], -3)]]
```

```
[76]: f = classifying_map_loops(torus, N, check=True); f
```

```
[76]: Simplicial set morphism:
        From: Torus
        To:   BG
        Defn: [(v_0, v_0), (s_0 v_0, sigma_1), (sigma_1, s_0 v_0), (sigma_1, sigma_1),
      (s_0 sigma_1, s_1 sigma_1), (s_1 sigma_1, s_0 sigma_1)] --> [(), 1, 3, 4, (1,
      3), (3, 1)]
```

Normal covering projection as pullback The following function implements all parts of algorithm 1 from the enclosed bachelor thesis in the version modified as in algorithm 4.

So, the function has input X, a finite connected simplicial set, and N, where N is a list of lists, $N = [n_0, \dots, n_s]$.

Each n_i is a list of tuples $(\sigma_{i_1}, k_{i_1}), \dots, (\sigma_{i_{r_i}}, k_{i_{r_i}})$ for each σ_{i_j} a nondegenerate 1-simplex of X and $k_{i_j} \in \mathbb{Z}$ such that $\sigma_{i_1}^{k_{i_1}} * \cdots * \sigma_{i_{r_i}}^{k_{i_{r_i}}}$ is forming a loop in the 1-skeleton of X.

The condition that n_i defines a loop says that

$$\text{for all } j = 1, \ldots, r_i \quad \begin{cases} d_0(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} > 0 \\ d_0(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} > 0 \text{ and } k_{i_{j+1 \bmod r_i}} < 0 \\ d_1(\sigma_{i_j}) = d_1(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j} < 0 \text{ and } k_{i_{j+1 \bmod r_i}} > 0 \\ d_1(\sigma_{i_j}) = d_0(\sigma_{i_{j+1 \bmod r_i}}) & \text{if } k_{i_j}, k_{i_{j+1 \bmod r_i}} < 0. \end{cases}$$

and $d_0(\sigma_{i_j}) = d_1(\sigma_{i_j})$ if $|k_{i_j}| > 1$.

If the function is called with the boolean check set to True, then it is check whether N is of the above form.

The function returns the pullback Z of the diagram $X \xrightarrow{f} BG \leftarrow EG$, where BG is the dim(X)-skeleton of the classifying space of G, EG is a universal cover of BG and f is a morphism of simplicial sets constructed in such a way that the following hold: 1) The canonical map $p: Z \to X$ is a covering projection. 2) Z is connected 3) $p_*(\pi_1(Z))$ is the subgroup of $\pi_1(X)$ generated by the elements of N 3) $\pi_1(f)$ is surjective.

[77]:
```
def NormalCoveringSpaceFromClassifyingSpaceLoops(X, N, check=True):
    #X a finite connected simplicial set
    #N loops in pi_1(X) generating a normal subgroup of fin. index
    f,BG,simpBG, G,G_list,card=classifying_map_normal_covering_loops(X,N,check)

    # construction of the dim X skeleton of EG and the covering map q: EG to BG

    simp = {(s,): AbstractSimplex(0, name='('+str(s)+')') for s in range(card)}
    #simp is a dictionary with keys given by tuples
    #each tuples has entries in range(card) and length leq dim X
    #the values of the dictionary simp are the simplices of EG

    q = {simp[(s,)]: simpBG[0] for s in range(0,card)}
    sSet={simp[(s,)]: None for s in range(0, card)}

    for n in range(1,X.dimension()+1):
        for s in product(range(0, card), repeat=(n+1)):
            if not any([s[i]==s[i+1] for i in range(0, n)]):
                # s gives non-degenerate of EG
                simp[s] = AbstractSimplex(n, name = str(s))
                faces = []
                for i in range(0, n+1):
                    if i>0 and i<n and s[i-1]==s[i+1]:
                        simplex = simp[s[0:i]+s[(i+2):len(s)]]
                        simplex = simplex.apply_degeneracies(i-1)
                        faces.append(simplex)
                    else:
                        faces.append(simp[s[0:i]+s[(i+1):len(s)]])
                sSet[simp[s]]=tuple(faces)
                g_s = [G_list[s[i]].inverse()*G_list[s[i+1]] for i in range(n)]
                q_s = [G_list.index(g) for g in g_s]
                q[simp[s]]=simpBG[tuple(q_s)]
    EG = SimplicialSet(sSet, name='EG', base_point=simp[(0,)])
    q_mor = SimplicialSetMorphism(q, EG, BG)
    return BG.pullback(f, q_mor)
```

Example: A triangulation of the torus as a Δ-complex with two 2-simplices: We compute a double cover of the torus by pullying back the universal covering projection $S^2 \to \mathbb{R}P^2$.

[78]:
```
torus=simplicial_sets.Torus()
```

[79]:
```
N=[[(torus.n_cells(1)[1], 1)], [(torus.n_cells(1)[0], 2)]]
```

[80]:
```
Z = NormalCoveringSpaceFromClassifyingSpaceLoops(torus, N)
```

[81]:
```
Z.homology()
```

[81]:
```
{0: 0, 1: Z x Z, 2: Z}
```

[82]:
```
Z.structure_map(0).induced_homology_morphism().to_matrix()
```

[82]:
```
[ 1| 0  0| 0]
[--+-----+--]
[ 0| 1 -1| 0]
[ 0| 0  2| 0]
[--+-----+--]
[ 0| 0  0| 2]
```

2 Join of simplicial sets and construction of lens spaces

2.0.1 Join of two simplicial sets

For later use we compute the join of simplicial sets.

[83]:
```
def join(s,t, simplices_non_distinct = True):
    if not (s.is_finite() and t.is_finite()):
        raise ValueError("the given simplicial sets might not be finite")

    # we force s,t to have no simplices in common
    if simplices_non_distinct:
        t = copy(t)
    #the naming of the simplices in this copy is not always as one expects
    simplices = {}
    data = {}
    for d in range(len(s.f_vector())+len(t.f_vector())+1): #dim of the join
        for i in range(-1, d+1):
            if i == -1:
                for simp in s.n_cells(d):
                    simplices[simp]=s.faces(simp)
            elif d-i-1 >= 0:
                for simp1 in s.n_cells(i):
                    for simp2 in t.n_cells(d-i-1):
                        name = str(simp1) + ' join ' +str(simp2)
                        simp = AbstractSimplex(d, name=name)
                        data[(simp1, simp2)] = simp
                        simp_faces = []
                        for k in range(d+1):
                            if  k <= i:
                                if i==0:
                                    simp_faces.append(simp2)
                                elif s.face(simp1,k).is_nondegenerate():
                                    simp_faces.append(data[(s.
↪face(simp1,k),simp2)])
                                else:
                                    f = s.face(simp1,k)
                                    underlying = f
                                    for l in f.degeneracies():
                                        underlying = s.face(underlying, l)
                                    new = AbstractSimplex(underlying.
↪dimension()+d-i, degeneracies=f.degeneracies(), underlying = data[(underlying,␣
↪simp2)])
                                    #new.dim() = underlying.dimension()+d-i
                                    simp_faces.append(new)
                            else:
                                if d-i-1==0:
                                    simp_faces.append(simp1)
                                elif t.face(simp2,k-i-1).is_nondegenerate():
                                    simp_faces.append(data[(simp1,t.
↪face(simp2,k-i-1))])
                                else:
                                    f= t.face(simp2,k-i-1)
                                    underlying = f
                                    for l in f.degeneracies():
                                        underlying = t.face(underlying, l)
                                    degeneracies = tuple([m + 1 + i for m in f.
↪degeneracies()])
                                    new = AbstractSimplex(i+underlying.
↪dimension()+1, degeneracies=degeneracies, underlying = data[(simp1,␣
↪underlying)])
                                    simp_faces.append(new)
                        simplices[simp]=tuple(simp_faces)
            elif d-i-1 == -1:
                 for simp in t.n_cells(d):
                    simplices[simp]=t.faces(simp)
    return SimplicialSet(simplices),  data
```

A very small example: We define S^5 to be the simplicial set with two nondegenerate simplices, where one of these two is a 5-simplex.

[84]:
```
s5 = simplicial_sets.Sphere(5)
```

The join $S^5 \star S^5$ should yield S^{11}, which has the following reduced homology:

[85]:
```
s11 = join(s5,s5)[0]; s11.homology()
```

[85]:
```
{0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: Z}
```

2.0.2 Join of morphisms of simplicial sets

For later use we also implement the join of morphisms of simplicial sets.

The following code needs the simplices of the codomains and the simplices of the domains to be distinct, respectively. this can be achieved by first using copy().

[86]:
```
def joinMorph(f, g ):
    d1 = f.domain()
    d2 = g.domain()
    c1 = f.codomain()
    c2 = g.codomain()
    if not (d1.is_finite() and d2.is_finite() and c1.is_finite() and c2.
↪is_finite()):
        raise ValueError("the given simplicial sets might not be finite")
    if (d1==c1 and d2 == c2):
        return joinEndo(f,g)
    C,  C_data = join(c1, c2, False)
    F = {}
    simplices = {}
    data = {}
    for d in range(len(d1.f_vector())+len(d2.f_vector())+1): #dim of the join
        for i in range(-1, d+1):
            if i == -1:
                for simp in d1.n_cells(d):
                    simplices[simp]=d1.faces(simp)
                    F[simp]=f(simp)
            elif d-i-1 >= 0:
                for simp1 in d1.n_cells(i):
                    for simp2 in d2.n_cells(d-i-1):
                        name = str(simp1) + ' join ' +str(simp2)
                        simp = AbstractSimplex(d, name=name)
                        data[(simp1, simp2)] = simp
                        F[simp]= C_data[(f(simp1), g(simp2))]
                        simp_faces = []
                        for k in range(d+1):
                            if  k <= i:
                                if i==0:
                                    simp_faces.append(simp2)
                                elif d1.face(simp1,k).is_nondegenerate():
                                    simp_faces.append(data[(d1.
↪face(simp1,k),simp2)])
                                else:
                                    f = d1.face(simp1,k)
                                    underlying = f
                                    for l in f.degeneracies():
                                        underlying = d1.face(underlying, l)
                                    new = AbstractSimplex(underlying.
↪dimension()+d-i, degeneracies=f.degeneracies(), underlying = data[(underlying,␣
↪simp2)])
                                    #new.dim=underlying.dimension()+d-i
                                    simp_faces.append(new)
                            else:
                                if d-i-1==0:
                                    simp_faces.append(simp1)
                                elif d2.face(simp2,k-i-1).is_nondegenerate():
                                    simp_faces.append(data[(simp1,d2.
↪face(simp2,k-i-1))])
                                else:
                                    f= d2.face(simp2,k-i-1)
                                    underlying = f
                                    for l in f.degeneracies():
                                        underlying = d2.face(underlying, l)
                                    degeneracies = tuple([m + 1 + i for m in f.
↪degeneracies()])
                                    new = AbstractSimplex(i+underlying.
↪dimension()+1, degeneracies=degeneracies, underlying = data[(simp1,␣
↪underlying)])
                                    simp_faces.append(new)

                        simplices[simp]=tuple(simp_faces)
            elif d-i-1 == -1:
                for simp in d2.n_cells(d):
                    simplices[simp]=d2.faces(simp)
                    F[simp]=g(simp)
    return SimplicialSetMorphism(F, SimplicialSet(simplices), C)
```

For better performance, we will also implement the join in the case that $d_1 = c_1$ and $d_2 = c_2$, separately.

[87]:
```
def joinEndo(f, g):
    d1 = f.domain()
    d2 = g.domain()
    c1 = f.codomain()
    c2 = g.codomain()
    if not (d1.is_finite() and d2.is_finite() and c1.is_finite() and c2.
↪is_finite()):
        raise ValueError("the given simplicial sets might not be finite")
    if not (d1==c1 and d2 == c2):
        raise ValueError("the domains don't equale the codomains")

    ## first building the simplicial set
    simplices = {}
    data = {}
    F = {} #the morphism
    for d in range(len(d1.f_vector())+len(d2.f_vector())+1): #the dimension of␣
↪the join
        for i in range(-1, d+1):
            if i == -1:
                for simp in d1.n_cells(d):
                    simplices[simp]=d1.faces(simp)
                    F[simp]=f(simp)
            elif d-i-1 >= 0:
                for simp1 in d1.n_cells(i):
                    for simp2 in d2.n_cells(d-i-1):
                        name = str(simp1) + ' join ' +str(simp2)
                        simp = AbstractSimplex(d, name=name)
                        data[(simp1, simp2)] = simp
                        simp_faces = []
                        for k in range(d+1):
                            if  k <= i:
                                if i==0:
                                    simp_faces.append(simp2)
                                elif d1.face(simp1,k).is_nondegenerate():
                                    simp_faces.append(data[(d1.
↪face(simp1,k),simp2)])
                                else:
                                    f = d1.face(simp1,k)
                                    underlying = f
                                    for l in f.degeneracies():
                                        underlying = d1.face(underlying, l)
                                    new = AbstractSimplex(underlying.
↪dimension()+d-i, degeneracies=f.degeneracies(), underlying = data[(underlying,␣
↪simp2)])
                                    simp_faces.append(new)
                            else:
                                if d-i-1==0:
                                    simp_faces.append(simp1)
                                elif d2.face(simp2,k-i-1).is_nondegenerate():
                                    simp_faces.append(data[(simp1,d2.
↪face(simp2,k-i-1))])
```

```
                                        else:
                                            f= d2.face(simp2,k-i-1)
                                            underlying = f
                                            for l in f.degeneracies():
                                                underlying = d2.face(underlying, l)
                                            degeneracies = tuple([m + 1 + i for m in f.
↪degeneracies()])
                                            new = AbstractSimplex(i+underlying.
↪dimension()+1, degeneracies=degeneracies, underlying = data[(simp1,␣
↪underlying)])
                                            simp_faces.append(new)

                                simplices[simp]=tuple(simp_faces)
                elif d-i-1 == -1:
                    for simp in d2.n_cells(d):
                        simplices[simp]=d2.faces(simp)
                        F[simp]=g(simp)
    #specifiying the morphism
        for i in range(0, d):
            for simp1 in d1.n_cells(i):
                for simp2 in d2.n_cells(d-i-1):
                    F[data[(simp1, simp2)]]=data[(f(simp1), g(simp2))]
    D = SimplicialSet(simplices)
    return SimplicialSetMorphism(F, D, D)
```

2.0.3 Construction of Lens Spaces

Z_p acts on S^1 by multiplication with ζ^{q_i} for ζ a p-th root of unity and $q_i \in \mathbb{Z}$. Choosing $q_1, \dots, q_n$ and defining $\rho_i : S^1 \to S^1, \rho_i(z) = \zeta^{q_i} \cdot z$, we obtain an action of $\mathbb{Z}_p$ on S^{2n-1} generated by $F = \rho_1 \star \cdots \star \rho_n : S^1 \star \cdots \star S^1 \to S^1 \star \cdots \star S^1$, noting that the n-fold join of S^1 is homeomorphic to S^{2n-1}.

The following function takes as input a positive integer $p \in \mathbb{Z}_{>0}$ and a list of integers $qs = [q_1, \dots, q_n]$. It returns the map $F : S^{2n-1} \to S^{2n-1}$ we described above. Further, the function returns the quotient map $q : S^{2n-1} \to S^{2n-1}/\mathbb{Z}_p$ to the orbits space of the action of $\mathbb{Z}_p = \mathbb{Z}/(p)$ on S^{2n-1} generated by F. The codomain of q is called a lens space and denoted $L(p; q_1, \dots, q_n)$ if and only if $\gcd(q_i, p) = q$ for all $i = 1, \dots, n$.

If $L(p; q_1, \dots, q_n)$ is a lens space, then q is a universal covering projection.

The function takes as input a boolean "check". If check is true, the function checks whether the given integers $q_1, \dots, q_n$ are all coprime to p and whether p is positive.

[88]:
```
def covLens(p,qs, check=True):
    if check:
        if not p>0:
            raise ValueError("p is not a positive integer")
        for q in qs:
            if not numpy.gcd(p,q) == 1:
                raise ValueError("the q's are not all coprime to p")
    v={}; e={}; S1s = {}; f={}
    for d in range(len(qs)):
        v[d] = [AbstractSimplex(0, name = 'v' +str(i)+'^('+str(d)+')') for i in␣
↪range(p)]
        e[d] = [AbstractSimplex(1, name = 'e' +str(i)+'^('+str(d)+')') for i in␣
↪range(p)]
        S1s[d] = SimplicialSet({e[d][i]: (v[d][i], v[d][(i+1)%p]) for i in␣
↪range(p)}, name = 'S^1_'+str(d))
        f[d] = SimplicialSetMorphism({e[d][i]: e[d][(i+qs[d])%p] for i in␣
↪range(p)}, S1s[d], S1s[d])
    F = f[0]
    for i in range(len(qs)-1):
        F=joinEndo(F, f[i+1])
    q = F.coequalizer(SimplicialSetMorphism(domain=F.domain(), codomain = F.
↪codomain(), identity=True))
    return F,q
```

Some lens spaces $F511 : S^3 \to S^3$ will be an automorphism of S^3 generating an action of $\mathbb{Z}/5$. $q511$ will be the coequalizer of the identity $S^3 \to S^3$ and $F511$. $F511$ was constructed in such a way, that $q511$ is the lens space $L(5;1,1)$. The structure map $S^3 \to q511$ of the coequalizer, will be the universal covering projection $S^3 \to L(5;1,1)$.

```
[89]: F511, q511 = covLens(5,[1,1])
```

```
[96]: F511.domain().homology() #reduced homology of S^3
```

```
[96]: {0: 0, 1: 0, 2: 0, 3: Z}
```

```
[97]: q511.homology() #reduced homoloty of L(5;1,1)
```

```
[97]: {0: 0, 1: C5, 2: 0, 3: Z}
```

Sometimes we are only interested in the lens space and not in the group action. So, we define a function that just returns the lens space.

The following function takes as input a positive integer $p \in \mathbb{Z}_{>0}$, a list of integers $qs = [q_1, \dots, q_n]$ and a boolean “check”. If check is true, the function checks whether the given integers $q_1, \dots, q_n$ are all coprime to p and whether p is positive.

The function returns a simplicial set, whose geometric realization is homeomorphic to $L(p; q_1, \dots, q_n)$.

```
[98]: def lens(p,qs, check=True):
          return SimplicialSet(covLens(p,qs, check)[1])
```

Let us compute the reduced homology of $L(4; -1, 3)$. Note that C4 denotes the cyclic group with four elements in SageMath.

```
[99]: lens(4, [-1,3]).homology()
```

```
[99]: {0: 0, 1: C4, 2: 0, 3: Z}
```

Let us construct a 5-dimensional lens space.

```
[100]: L4_135 = lens(4,[1,-1,5])
```

We compute the reduced homology of $L(4;1,-1,5)$.

```
[101]: L4_135.homology()
```

```
[101]: {0: 0, 1: C4, 2: 0, 3: C4, 4: 0, 5: Z}
```

Let us compute the normalized chain complex of S^3 by giving its matrices with respect to a $\mathbb{Z}[C5]$-basis' of its chain groups. The basis elements are given by lifts of nondegenerate simplices of $L(5;1,3)$ along the covering projection $S^3 \to L(5;1,3)$. Here $C5$ denotes the cyclic group with five elements.

```
[102]: C_L513=complexOfUniversalCover(lens(5, [1,3]), True)
```

```
[103]: C_L513[1].base_ring()
```

```
[103]: Algebra of Finitely presented group < e0 | e0^5 > over Integer Ring
```

We print "C_L513" in LaTeX:

$$\left\{ 1 : \begin{pmatrix} -1+e_0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 0 & -1+e_0^2 & 1 & e_0^{-2} & e_0 & e_0^{-1} & e_0^2 \end{pmatrix}, \quad 2 : \left(\begin{array}{cccccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & e_0^{-2} & e_0 & e_0^{-1} & e_0^2 & \\ e_0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & \\ 0 & e_0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & \\ -1 & 0 & e_0 & 0 & 0 & 0 & 1 & -1 & 0 & \\ 0 & -1 & 0 & e_0 & 0 & 0 & 0 & 1 & -1 & \\ 0 & 0 & -1 & 0 & e_0 & 0 & 0 & 0 & 1 & - \end{array} \right. \right.$$

Let us compute the normalized chain complex of S^5 by giving its matrices with respect to a $\mathbb{Z}[C4]$-basis' of its chain groups. The basis elements are given by lifts of nondegenerate simplices of $L(4;1,-1,5)$ along the covering projection $S^5 \to L(4;1,-1,5)$. Here $C4$ denotes the cyclic group with five elements.

```
[105]: C_L4_135=complexOfUniversalCover(L4_135, True)
```

```
[106]: C_L4_135[1].base_ring()
```

```
[106]: Algebra of Finitely presented group < e0 | e0^4 > over Integer Ring
```

3 Reidemeister Torsion

3.0.1 Construction of the twist complex

The following function implements algorithm 8 of the enclosed bachelor thesis but only returns the twist complex not the Reidemeister torsion.

The function takes as input a connected, finite simplicial set X, as well as, list of edges $gens = [e_1, \ldots, e_m]$ in X such that $\mathrm{sk}_1(X) \setminus \{e_1, \ldots, e_m\}$ is a spanning tree in the 1-skeleton of X.

Secondly, the function takes a list of matrices $\phi = \{M_1, \ldots, M_m\}$, which are all complex square matrices of the same size such that sending $e_i \mapsto M_i$ induces a group homomorphism $\pi_1(X) \to \mathrm{GL}(n, \mathbb{C})$. Here we use a presentation of $\pi_1(X)$ with generators $e_1, \ldots, e_m$ and relations given by the 2-simplices of X.

The function takes as input a boolean “check”, which is by default set True. If “check” is set True, then the algorithm checks, whether those conditions above are satisfied.

The function returns the chain complex $C(X, \phi)$ of finite dimensional $\mathbb{C}$-vector spaces defined in definition 3.15 of the enclosed bachelor thesis by giving the matrices of the boundary operators with respect to the distinguished basis.

[117]:
```
def twistComplex(X, gens, phi, check=True):

    #X: finite connected simplicial set,
    #gens: edges not contained in fixed spanning tree of sk_1(X)
    #phi: lists of matrices assigned to the gens,
    #such that e[i] -> phi[i] gives representation of p_1(X)
    #if Check, then we check if phi really is a representation

    if not X.is_finite():
        raise ValueError("given simplicial set might not be finite")
    f_vec = X.f_vector()
    if len(f_vec)<2:
        return X.chain_complex()
    if not X.is_connected():
        raise ValueError("given simplicial set is not connected")
    if len(gens)!=len(phi):
        raise ValueError("len(gens)!=len(phi)")

    #check if all matrices have the same size:
    if len(phi)>0:
        n = phi[0].dimensions()[0]
    else:
        n = 1
    for i in range(len(gens)):
        if not (n==phi[i].dimensions()[0] and n==phi[i].dimensions()[1]):
            raise ValueError('given matrizes are not of the same size')

    #for later extensive  use
    I = matrix.identity(n,CDF)

    #next we check whether sk_1(X)-gens is a spanning tree in sk_1(X)
    if check:
        treeSimplices = {}
        for e in [item for item in X.n_cells(1) if item not in gens]:
            treeSimplices[e]=(X.face(e,0), X.face(e,1))
        for v in X.n_cells(0):
            treeSimplices[v]=None
        tree = SimplicialSet(treeSimplices)
        if not tree.is_connected():
            raise ValueError("sk_1(X)-gens not spanning subgraph of sk_1(X)")
        if not tree.is_acyclic():
            raise ValueError("sk_1(X)-gens is not acyclic")

        #next we check if phi really is a homomorphism,
        #so we check wheter the relations in pi_1(X) are satisfied
        for f in X.n_cells(2):
            z = dict()
            for i in range(3):
                e = X.face(f,i)
                if e.is_degenerate():
                    z[i]=matrix.identity(n)
                elif e in gens:
                    z[i]=phi[gens.index(e)]
                else:
                    z[i]=matrix.identity(n)
            if not numpy.allclose(z[0]*z[1].inverse()*z[2], I):
                raise ValueError("phi is not a homomorphism")

    deckTrnsf = {i: [] for i in range(1, len(f_vec))}
    twistComplx = {}
    for i in range(1, len(f_vec)):
        twistComplx[i] = matrix(CDF, f_vec[i-1]*n, f_vec[i]*n)
    #we start with 0 matrices and fill their entries later



    for j in range(f_vec[1]):
        e = X.n_cells(1)[j]
        ind_d0 = X.n_cells(0).index(X.face(e,0))
        ind_d1 = X.n_cells(0).index(X.face(e,1))
        twistComplx[1][n*ind_d1:n*(ind_d1+1),n*j:n*(j+1)] -= I
        if e in gens:
            i = gens.index(e)
            twistComplx[1][n*ind_d0:n*(ind_d0+1), n*j:n*(j+1)] +=phi[i]
            deckTrnsf[1].append(phi[i])
        else:
            twistComplx[1][n*ind_d0:n*(ind_d0+1),n*j: n*(j+1)] += I
            deckTrnsf[1].append(I)

    for k in range(2, len(f_vec)):
        for j in range(f_vec[k]):
            f = X.n_cells(k)[j]
            for m in range(1, k+1):
                if X.face(f,m).is_nondegenerate():
                    i=X.n_cells(k-1).index(X.face(f,m))
                    twistComplx[k][n*i: n*(i+1),n*j:n*(j+1)] += (-1)^m*I
            dk = X.face(f,k)
            if dk.is_degenerate() and 0 in dk.degeneracies():
                if X.face(f,0).is_nondegenerate():
                    i= X.n_cells(k-1).index(X.face(f,0))
                    twistComplx[k][n*i: n*(i+1),n*j:n*(j+1)] += I
                deckTrnsf[k].append(I)
            else:
                dim_und = dk.nondegenerate().dimension()
                ind_und = X.n_cells(dim_und).index(dk.nondegenerate())
                phi_g_0 = deckTrnsf[dim_und][ind_und]
                if X.face(f,0).is_nondegenerate():
                    i = X.n_cells(k-1).index(X.face(f,0))
                    twistComplx[k][n*i: n*(i+1),n*j:n*(j+1)] += phi_g_0
                deckTrnsf[k].append(phi_g_0)
    return twistComplx
```

3.0.2 Torsion of bounded, based, finite dimensional $\mathbb{C}$-vector space chain complexes

The following function implements algorithm 7 of the enclosed bachelor thesis.

The function has input a dictionary C. The keys of that dictionary are integers indexing the nonzero chain groups of a bounded chain complex of finite dimensional $\mathbb{C}$-vector spaces. The value of the dictionary at k is the matrix $D^{(k)}$ that represents the boundary operator $\partial_k : C_k \to C_{k-1}$ of that chain complex with respect to some distinguished basis.

The matrices need to have coefficients compatible with CDF, the complex double field:

```
[109]: ComplexDoubleField() is CDF
```

```
[109]: True
```

The function returns the torsion of the chain complex, based with the distinguished basis, as a complex number $\mathbb{C}$. The function returns $0 \in \mathbb{Z}$ if and only if C is not exact.

The function does not check, whether C is a complex.

```
[111]: def torsion_complex(C):
           #C a dictionary with keys a bounded interval of ZZ
```

```
    #the values of C are matrices such that C[k]C[k+1]=0
    t = 1 #the torsion is initially set to 1
    d= max(C) #the upper bound of the complex C
    low_bound=min(C)-1 #lower bound of the complex C

    r = {d: 0, d-1: C[d].dimensions()[1]}
    #if C is exact, then r[k-1]=rank(C[k]) for all k

    K =matrix.identity(r[d-1], CDF)
    for k in range(d, low_bound,-1):
        if r[k-1]>C[k].dimensions()[0]:
            return 0
        r[k-2]=C[k].dimensions()[0]-r[k-1]
        if r[k-1]==0:
            K=matrix.identity(r[k-2], CDF)
            continue
        U,S,V=C[k].SVD()

        #check if the r[k-1]'th singular value of C[k] is 0
        if numpy.isclose(0, S[r[k-1]-1,r[k-1]-1]):
            return 0
        detW=(V.conjugate_transpose()*K).submatrix(0,0,r[k-1],r[k-1]).det()
        t=t*(U.det()*prod(S[j,j] for j in range(r[k-1]))*detW)^((-1)^k)
        K=U.submatrix(0, r[k-1], C[k].dimensions()[0], r[k-2])

    #check whether we are exact at C[low_bound]
    if r[low_bound-1]==0:
        return t
    else:
        return 0
```

3.0.3 Reidemeister torsions for group presentations provided by the user

Example 1: The lens space $L(5;1,1)$. We start by computing the torsion of a 3-dimensional lens space.

[112]:
```
L511=lens(5,[1,1])
```

[113]:
```
G, gens =fundamental_group_w_gens(L511); G.simplification_isomorphism()
```

[113]:
```
Generic morphism:
  From: Finitely presented group < e0, e1, e2, e3, e4, e5 | e4^-1*e0, e1*e0,
e2*e1^-1*e0, e3*e2^-1*e0, e4*e3^-1*e0, e5*e1, e5*e1^-1*e2, e5*e2^-1*e3,
e5*e3^-1*e4, e5*e4^-1 >
  To:   Finitely presented group < e0 | e0^5 >
  Defn: e0 |--> e0
        e1 |--> e0^-1
```

```
e2 |--> e0^-2
e3 |--> e0^2
e4 |--> e0
e5 |--> e0
```

```
[114]: W = Matrix(CDF,[numpy.exp(2*numpy.pi*I/5)])
```

We construct ϕ to be the presentation of $\pi_1(L(5;1,1))$ that send e_0 to W.

```
[115]: phi = [W, W^(-1), W^(-2), W^2, W, W]
```

```
[118]: C= twistComplex(L511, gens ,phi, True)
```

We check, whether C is a chain complex.

```
[119]: is_C_a_chain_complex=[]
       for i in range(1, L511.dimension()):
           is_C_a_chain_complex.append(numpy.allclose(C[i]*C[i+1],0))
       is_C_a_chain_complex
```

```
[119]: [True, True]
```

Now let us compute the torsion of C:

```
[120]: torsion_complex(C)
```

```
[120]: 0.22360679774997927 - 0.6881909602355871*I
```

Let us compare the computed number, with the theoretical result given in theorem 10.6 in turaev's introduction to topological torsion:

```
[121]: z=numpy.exp(2*numpy.pi*I/5); tors_turaev=-(z-1)^(-1)*(z-1)^(-1); tors_turaev
```

```
[121]: (0.22360679774997905-0.688190960235587j)
```

In other words, we computed the correct result, up to the 16th digit, which is as close we can get using double precision floats.

```
[122]: tors_turaev-torsion_complex(C)
```

```
[122]: (-2.220446049250313e-16+1.1102230246251565e-16j)
```

Example 2: The lens space L(4;1,3,1)

```
[123]: L4131=lens(4, [1,3,1])
```

```
[124]: G, gens=fundamental_group_w_gens(L4131); G.simplification_isomorphism()
```

[124]:
```
Generic morphism:
  From: Finitely presented group < e0, e1, e2, e3, e4, e5, e6, e7, e8, e9, e10,
e11, e12 | e1*e2^-1*e0, e7^-1*e0, e2*e3^-1*e0, e5*e0, e3*e4^-1*e0, e6*e5^-1*e0,
e4*e1^-1*e0, e7*e6^-1*e0, e9^-1*e8, e9*e10^-1*e8, e10*e11^-1*e8, e11*e8,
e8*e1^-1*e2, e12*e5, e8*e2^-1*e3, e12*e5^-1*e6, e8*e3^-1*e4, e12*e6^-1*e7,
e8*e4^-1*e1, e12*e7^-1, e1, e9*e5^-1*e1, e10*e6^-1*e1, e11*e7^-1*e1, e9*e2,
e10*e5^-1*e2, e11*e6^-1*e2, e7^-1*e2, e10*e3, e11*e5^-1*e3, e6^-1*e3,
e9*e7^-1*e3, e11*e4, e5^-1*e4, e9*e6^-1*e4, e10*e7^-1*e4, e12*e9, e12*e9^-1*e10,
e12*e10^-1*e11, e12*e11^-1 >
  To:   Finitely presented group < e0 | e0^4 >
  Defn: e0 |--> e0
        e1 |--> 1
        e2 |--> e0
        e3 |--> e0^-2
        e4 |--> e0^-1
        e5 |--> e0^-1
        e6 |--> e0^-2
        e7 |--> e0
        e8 |--> e0^-1
        e9 |--> e0^-1
        e10 |--> e0^-2
        e11 |--> e0
        e12 |--> e0
```

[125]:
```
w = Matrix(CDF,[numpy.exp(2*numpy.pi*I/4)])
```

We construct a presentation ϕ of $\pi_1(L(4;1,3,1))$ sending e_0 to w.

[126]:
```
phi = [w,w^0,w,w^(-2),w^(-1),w^(-1),w^(-2),w,w^(-1),w^(-1),w^(-2),w,w ]
```

[127]:
```
Cw= twistComplex(L4131, gens ,phi, True)
```

Again we check if the algorithm worked and we constructed a chain complex.

[128]:
```
is_Cw_a_chain_complex=[]
for i in range(1, L4131.dimension()):
    is_Cw_a_chain_complex.append(numpy.allclose(Cw[i]*Cw[i+1],0))
is_Cw_a_chain_complex
```

[128]:
```
[True, True, True, True]
```

Let us compute the torsion of this chain complex with respect to the distinguished (standard) basis.

[129]:
```
torsion_complex(Cw)
```

[129]:
```
-0.24999999999999933 - 0.24999999999999967*I
```

If we compare the computed result with the theoretical result computed in Turaev's book on topological torsion we obtain, we see that we again computed the correct result up to machine precision.

```
[130]: theoretical_result = -(1-w)^(-1)*(1-w^3)^(-1)*(1-w)^(-1); theoretical_result
```

```
[130]: [-0.24999999999999994 - 0.2500000000000001*I]
```

```
[131]: matrix(CDF, [torsion_complex(Cw)])-theoretical_result
```

```
[131]: [6.106226635438361e-16 + 4.440892098500626e-16*I]
```

Example 3: A 2-dimensional presentation of $\pi_1(L(5;1,1))$

```
[132]: L511=lens(5,[1,1])
```

```
[133]: G, gens =fundamental_group_w_gens(L511); G.simplification_isomorphism()
```

```
[133]: Generic morphism:
         From: Finitely presented group < e0, e1, e2, e3, e4, e5 | e4^-1*e0, e1*e0,
       e2*e1^-1*e0, e3*e2^-1*e0, e4*e3^-1*e0, e5*e1, e5*e1^-1*e2, e5*e2^-1*e3,
       e5*e3^-1*e4, e5*e4^-1 >
         To:   Finitely presented group < e0 | e0^5 >
         Defn: e0 |--> e0
               e1 |--> e0^-1
               e2 |--> e0^-2
               e3 |--> e0^2
               e4 |--> e0
               e5 |--> e0
```

```
[134]: W = Matrix(CDF,[[numpy.exp(2*numpy.pi*I/5), 0], [0,numpy.exp(-2*numpy.pi*I/5)]])
```

We construct ϕ to be the presentation of $\pi_1(L(5;1,1))$ that send e_0 to the two dimensional matrix W.

```
[135]: phi = [W, W^(-1), W^(-2), W^2, W, W]
```

```
[136]: CW2= twistComplex(L511, gens, phi, True)
```

We check again, that twistComplex indeed gives a chain comlex, even though we know this by the theory.

```
[137]: is_CW2_a_chain_complex=[]
       for i in range(1, L511.dimension()):
           is_CW2_a_chain_complex.append(numpy.allclose(CW2[i]*CW2[i+1],0))
       is_CW2_a_chain_complex
```

```
[137]: [True, True]
```

Let us compute the torsion of this chain complex, which is an element of $\mathbb{C}^*/\pm$, because W has determinant 1.

```
[138]: torsion_complex(CW2)
```

```
[138]: 0.5236067977499784 + 5.551115123125783e-17*I
```

We compare this result with lemma 3.24 of the enclosed bachelor thesis.

```
[139]: z3=numpy.exp(2*numpy.pi*I/5);
```

```
[140]: t_L_1 = (z3-1)^(-1)*(z3-1)^(-1); t_L_2 = (z3^(-1)-1)^(-1)*(z3^(-1)-1)^(-1)
```

```
[141]: torsion_complex(CW2)-(t_L_1)*(t_L_2)
```

```
[141]: -6.661338147750939e-16 + 1.1102230246251565e-16*I
```

The number above is zero, up to machine precision.

Example 3: Some Reidemeister torsions of $\pi_1(L(5;2,3))$

```
[142]: L523=lens(5,[2,3])
```

```
[143]: G, gens = fundamental_group_w_gens(L523)
```

```
[144]: G.simplification_isomorphism()
```

```
[144]: Generic morphism:
          From: Finitely presented group < e0, e1, e2, e3, e4, e5 | e1^-1*e0,
        e1*e2^-1*e0, e2*e3^-1*e0, e3*e4^-1*e0, e4*e0, e5*e1, e5*e1^-1*e2, e5*e2^-1*e3,
        e5*e3^-1*e4, e5*e4^-1 >
          To:   Finitely presented group < e0 | e0^5 >
          Defn: e0 |--> e0
                e1 |--> e0
                e2 |--> e0^2
                e3 |--> e0^-2
                e4 |--> e0^-1
                e5 |--> e0^-1
```

```
[145]: zeta_5 = numpy.exp(2*4*numpy.pi*I/5)
```

```
[146]: W = Matrix(CDF,[[zeta_5, 0], [0,zeta_5^(-1)]])
```

```
[147]: C523= twistComplex(L523, gens ,[W, W, W^(2), W^(-2), W^(-1), W^(-1)], True)
```

```
[148]: torsion_complex(C523)
```

```
[148]: 0.5236067977499772 - 4.787836793695988e-16*I
```

```
[149]: w = Matrix(CDF,[zeta_5]); z = Matrix(CDF,[zeta_5^(-1)])
```

```
[150]: C523w= twistComplex(L523, gens ,[w, w, w^(2), w^(-2), w^(-1), w^(-1)], True)
       C523z= twistComplex(L523, gens ,[z, z, z^(2), z^(-2), z^(-1), z^(-1)], True)
```

```
[151]: torsion_complex(C523w), torsion_complex(C523z)
```

```
[151]: (-0.7236067977499787 - 8.326672684688674e-16*I,
        -0.7236067977499787 + 8.326672684688674e-16*I)
```

Again we compare this with the theoretical result and lemma 3.24.

```
[152]: -(w-1)^(-1)*(w^(-1)-1)^(-1)
```

```
[152]: [-0.7236067977499788]
```

```
[153]: ((w-1)^(-1)*(w^(-1)-1)^(-1))^2
```

```
[153]: [0.5236067977499788]
```

3.0.4 Reidemeister torsions with respect to all irreducible representations

The following function has input a finite connected simplicial set X with finite fundamental group G.

The function returns a list of complex numbers.

The entries of the list are in bijection to the irreducible complex representations of G. For every irreducible representation ϕ of G the corresponding entry in the list is the Reidemeister torsion of X with resepct to ϕ, i.e.

$$\text{all\_r\_torsions}(X) = [\tau(X, \phi) \text{ for } \phi \text{ irrep of } G].$$

The algorithm has, additionally to X, input a boolean called "simplification". If "simplficiation" is set True, then we first simplify a computed presentation of G and then use GAP to compute irreducible representations. If "simplification" is set False, then GAP is immediately asked to compute irreducible representatins. Setting simplification to be True can have the advantage of better performance.

```
[154]: def all_r_torsions(X, simplification = True):

           G, gens = fundamental_group_w_gens(X)
           if not X.is_finite():
               raise ValueError("the given simplicial set might not be finite")
           if not G.is_finite():
               raise ValueError("fundamental group of X might be infinite")

           if not simplification:
               all_irreps = G.gap().IrreducibleRepresentations()
               n_irr = len(all_irreps)
```

```
        # we build a dictionary of all the irreducible representations
        phi_dict = {}
        for i in range(n_irr):
            phi_dict[i]=[]
            for j in range(len(gens)):
                phi_j = all_irreps[i].Image(G.gen(j))
                phi_dict[i].append(Matrix(CDF, phi_j.sage()))

    if simplification:
        h = G.simplification_isomorphism()
        G_simp = h.codomain()
        len_gens_si = len(G_simp.gens())
        if len_gens_si==0:
            return [0]
        irreps_G_simp = G_simp.gap().IrreducibleRepresentations()
        n_irr = len(irreps_G_simp)
        # we build a dictionary of all the irreducible representations
        phi_dict = {i: [] for i in range(n_irr)}
        for i in range(n_irr):
            irrep = [] #the i'th irrep
            for j in range(len_gens_si):
                phi_j = irreps_G_simp[i].Image(G_simp.gen(j))
                irrep.append(Matrix(CDF, phi_j.sage()))
            n= irrep[0].dimensions()[0]
            for j in range(len(gens)):
                e_titz = h(G.gen(j)).Tietze()
                rep_e = matrix.identity(n, CDF)
                for m in range(len(e_titz)):
                    rep_e = rep_e*irrep[abs(e_titz[m])-1]^(sgn(e_titz[m]))
                phi_dict[i].append(rep_e)

    t_list = [] #the list of torsions w.r.t every irrep

    #first we fill the list with the twist complexes
    for i in range(n_irr):
        t_list.append(twistComplex(X, gens, phi_dict[i], False))

    #then we compute the torsions of these complexes
    for i in range(n_irr):
        t_list[i]=torsion_complex(t_list[i])


    return t_list
```

Reidemeister torsions of lens spaces We compute the Reidemeister torsions of a bunch of lens spaces and compare with the theoretical result. The theoretical result can, for example, be found in Turaev's Introduction to Topological Torsion.

[155]:
```
all_r_torsions(lens(5,[1,1]))
```

[155]:
```
[0,
 0.22360679774997927 - 0.6881909602355871*I,
 -0.22360679774997883 - 0.16245984811645317*I,
 -0.2236067977499793 + 0.16245984811645342*I,
 0.22360679774997924 + 0.6881909602355878*I]
```

[156]:
```
zeta = numpy.exp(2*numpy.pi*I/5); (zeta-1)^(-1)*(zeta-1)^(-1)
```

[156]:
```
(-0.22360679774997905+0.688190960235587j)
```

[157]:
```
all_r_torsions(lens(5,[1,2]))
```

[157]:
```
[0,
 -0.1381966011250111 - 0.4253254041760213*I,
 -0.36180339887498825 + 0.262865556059566*I,
 -0.36180339887499047 - 0.2628655560595675*I,
 -0.13819660112501148 + 0.42532540417602194*I]
```

[159]:
```
-(zeta-1)^(-1)*(zeta^3-1)^(-1)
```

[159]:
```
(-0.3618033988749896-0.26286555605956685j)
```

[160]:
```
all_r_torsions(lens(5,[1,2]), simplification=False)
```

[160]:
```
[0,
 -0.36180339887498736 + 0.262865556059565*I,
 -0.13819660112501142 + 0.42532540417602216*I,
 -0.1381966011250101 - 0.4253254041760185*I,
 -0.3618033988749911 - 0.26286555605956785*I]
```

[161]:
```
all_r_torsions(lens(7,[1,1]))
```

[161]:
```
[0,
 0.827985277605681 + 1.0382606982861646*I,
 -0.09100904850610321 + 0.39873669444120063*I,
 -0.23697622909957736 + 0.11412173719507447*I,
 -0.23697622909957808 - 0.1141217371950752*I,
 -0.09100904850610339 - 0.3987366944412012*I,
 0.8279852776056815 - 1.0382606982861664*I]
```

[162]:
```
-(1-numpy.exp(2*numpy.pi*I/7))^(-2)
```

[162]:
```
(0.8279852776056821-1.0382606982861684j)
```

[163]:
```
all_r_torsions(lens(7,[1,2]))
```

```
[163]: [0,
        0.16399263880283999 + 0.7184986963636819*I,
        -0.2955045242530504 + 0.14230747862306287*I,
        -0.36848811454978864 - 0.4620694805455461*I,
        -0.3684881145497897 + 0.4620694805455484*I,
        -0.29550452425305146 - 0.14230747862306328*I,
        0.16399263880284015 - 0.7184986963636808*I]
```

```
[164]: all_r_torsions(simplicial_sets.RealProjectiveSpace(3))
```

```
[164]: [0, 0.25]
```

```
[165]: L4_113 = lens(4,[1,1,3]); L4_113
```

```
[165]: Simplicial set with 182 non-degenerate simplices
```

```
[166]: all_r_torsions(L4_113)
```

```
[166]: [0,
        -0.12499999999999892,
        -0.24999999999999659 - 0.24999999999999734*I,
        -0.24999999999999653 + 0.2499999999999969*I]
```

```
[167]: (numpy.exp(2*numpy.pi*I/4)-1)^(-2)*(numpy.exp(2*numpy.pi*I/4)^3-1)^(-1)
```

```
[167]: (-0.24999999999999997-0.2500000000000001j)
```

All Reidemeister Torsions of the Poincaré homology 3-sphere:

```
[168]: P_S3 = SimplicialSet(simplicial_complexes.PoincareHomologyThreeSphere())
       P_S3
```

```
[168]: Simplicial set with 392 non-degenerate simplices
```

The used triangulation of the Poincaré homology sphere as a simplicial complex has 392 simplices. The fundamental group of the Poincaré homology sphere has 120 elements.

```
[170]: all_r_torsions(P_S3)
```

```
[170]: [0,
        0.927888567837343 - 0.535535467990794*I,
        -0.09373507768770462 - 0.1025408020294057*I,
        0.0010993356262109098 - 5.827586677109586e-19*I,
        -0.006093620732141847 - 3.528535970354074e-16*I,
        0.0007567405205541789 - 2.4940988397388446e-05*I,
        -0.013553909859376267 - 0.00144350448822123*I,
        -0.0009875426507273146,
        4.549675882629726e-05 + 7.864640689903552e-05*I]
```

Example: Invariance of Reidemeister torsion under subdivision We use two different simplicial sets, which both have geometric realization homeomorphic to S^2. S2, below, has only two nondegenerate simplicies. S2_2, below, is a minimal triangulation of S^2 as a simplicial complex.

```
[171]: S2=simplicial_sets.Sphere(2);
       S2_2 = SimplicialSet(simplicial_complexes.Sphere(2));
       RP3=simplicial_sets.RealProjectiveSpace(3)
```

```
[172]: S2xRP3 = S2.product(RP3); S2_2xRP3 = S2_2.product(RP3)
```

```
[173]: all_r_torsions(S2xRP3)
```

```
[173]: [0, 0.0624999999999999875]
```

```
[174]: all_r_torsions(S2_2xRP3)
```

```
[174]: [0, 0.0625000000000000044]
```

Reidemeistertorsion of products: We compute the Reidemeister torsions of some examples of products of simplicial sets.

```
[175]: L1 = lens(3,[1,1])
       L2 = lens(3,[1,2])
       S2=simplicial_sets.Sphere(2)
       CP2=simplicial_sets.ComplexProjectiveSpace(2)
       RP2=simplicial_sets.RealProjectiveSpace(2)
       RP3=simplicial_sets.RealProjectiveSpace(3)
```

The Reidemeister torsion of $S^2 \times \mathbb{R}P^3$ is the Reidemeister torsion of $\mathbb{R}P^3$ squared.

```
[176]: all_r_torsions(S2.product(RP3)), 0.25^2
```

```
[176]: ([0, 0.0624999999999999875], 0.0625000000000000)
```

Let us look at $L1 = L(3;1,1)$. The first number is the theoretical result.

```
[180]: zeta = numpy.exp(2*numpy.pi*I/3)
       -(1-zeta)^(-2)
```

```
[180]: (-0.16666666666666669-0.2886751345948129j)
```

```
[181]: all_r_torsions(L1)
```

```
[181]: [0,
        -0.166666666666666652 - 0.288675134594812287*I,
        -0.166666666666666674 + 0.288675134594813325*I]
```

The Reidemeister torsion of $L(3;1,1) \times \mathbb{R}P^2$ equals the Reidemeister torsion of $L(3;1,1)$.

```
[182]: L1xRP2 = SimplicialSet(L1.product(RP2)); all_r_torsions(L1xRP2)
```

```
[182]: [0,
 0,
 0.16666666666666521 + 0.28867513459481203*I,
 0.16666666666666613 + 0.2886751345948118*I,
 0.1666666666666637 - 0.28867513459480926*I,
 0.16666666666667385 - 0.2886751345948277*I]
```

The Reidemeister torsion of $L(3;1,1) \times S^2$ equals the Reidemeister torsion of $L(3;1,1)$ squared.

```
[183]: L1xS2 = SimplicialSet(L1.product(S2)); all_r_torsions(L1xS2)
```

```
[183]: [0,
 -0.05555555555555479 + 0.09622504486493601*I,
 -0.0555555555555545 - 0.09622504486493554*I]
```

```
[184]: -((1-zeta)^(-2))^2
```

```
[184]: (0.05555555555555557-0.09622504486493764j)
```

Now we look at $L2 = L(3;1,2)$. The first number below is the theoretical result for the Reidemeister torsion of $L(3;1,2)$.

```
[185]: -(zeta-1)^(-1)*(zeta^2-1)^(-1)
```

```
[185]: (-0.33333333333333326-1.1102230246251565e-16j)
```

```
[186]: all_r_torsions(L2)
```

```
[186]: [0,
 -0.3333333333333331 - 1.1102230246251565e-16*I,
 -0.3333333333333333 - 2.220446049250313e-16*I]
```

The Reidemeister torsion of $L(3;1,2) \times \mathbb{R}P^2$ equals the Reidemeister torsion of $L(3;1,2)$.

```
[187]: L2xRP2=SimplicialSet(L2.product(RP2)); all_r_torsions(L2xRP2)
```

```
[187]: [0,
 0,
 0.3333333333333321 + 6.938893903907228e-16*I,
 0.3333333333333428 - 1.249000902703301e-16*I,
 0.33333333333334264 - 5.48172618408671e-16*I,
 0.33333333333332804 - 5.551115123125783e-16*I]
```

The Reidemeister torsion of $L(3;1,2) \times S^2$ equals the Reidemeister torsion of $L(3;1,2)$ squared

```
[188]: L2xS2=SimplicialSet(L2.product(S2)); tors_L2xS2 = all_r_torsions(L2xS2);␣
       ↪tors_L2xS2
```

```
[188]: [0,
        0.1111111111111119 + 2.7755575615628914e-17*I,
        0.11111111111111346 + 3.8163916471489756e-17*I]
```

```
[189]: (-(zeta-1)^(-1)*(zeta^2-1)^(-1))^2
```

```
[189]: (0.11111111111111106+7.401486830834375e-17j)
```

Now let us look at the Reidemeister torsion of the product of $L(3;1,1) \times L(3;1,2)$.

```
[190]: L1xL2 = SimplicialSet(L1.product(L2)); L1xL2
```

```
[190]: Simplicial set with 2580 non-degenerate simplices
```

So, we test our algorithm for all Reidemeister torsions of a space with 2580 nondegenerate simplices:

```
[191]: r_torsions_L1xL2=all_r_torsions(L1xL2)
```

```
[192]: r_torsions_L1xL2
```

```
[192]: [0,
        -1.0000000000000013 + 1.6403545188836688e-14*I,
        -0.9999999999999954 - 1.3100631690576847e-14*I,
        -1.0000000000002252 + 1.7486012637846216e-14*I,
        -1.0000000000000349 - 2.3314683517128287e-15*I,
        -1.0000000000000482 - 2.1815882433884326e-14*I,
        -0.999999999999917 + 1.3795605283140056e-14*I,
        -1.0000000000001148 - 9.131584377541913e-15*I,
        -0.999999999999919 - 2.914335439641036e-14*I]
```

So, the Reidemeister torsion of $L(3;1,1) \times L(3;1,2)$ is one.

Now, we look at the product $L(3;1,2) \times \mathbb{C}P^2$. Note that $\mathbb{C}P^2$ has Euler characteristic equal to 3.

```
[193]: L2xCP2=SimplicialSet(L2.product(CP2)); L2xCP2
```

```
[193]: Simplicial set with 3544 non-degenerate simplices
```

```
[194]: all_r_torsions(L2xCP2)
```

```
[194]: [0,
        0.03703703703703259 - 6.555086334847366e-16*I,
        0.03703703703704223 + 3.434752482434078e-16*I]
```

The torsion of $L(3;1,2) \times \mathbb{C}P^2$ is the torsion of $L(3;1,2)$ cubed:

```
[195]: ((zeta-1)^(-1)*(zeta^2-1)^(-1))^3
```

```
[195]: (0.037037037037037014+3.700743415417187e-17j)
```

The torsion of $\mathbb{R}P^3 \times \mathbb{C}P^2$ is the torsion of $\mathbb{R}P^3$ cubed:

```
[196]: RP3xCP2=SimplicialSet(RP3.product(CP2)); RP3xCP2
```

```
[196]: Simplicial set with 920 non-degenerate simplices
```

```
[197]: all_r_torsions(RP3xCP2), 0.25^3
```

```
[197]: ([0, 0.015624999999999424], 0.0156250000000000)
```

All computed examples stick to the formula: Let X, Y be CW-complexes and $\phi : \pi_1(X) \to \mathrm{GL}(\mathbb{C}^n)$, $\psi : \pi_1(Y) \to \mathrm{GL}(\mathbb{C}^m)$ representations of their fundamental groups. Then

$$\tau(X \times Y, \phi \otimes \psi) = \tau(X, \phi)^{\chi(Y)} \tau(Y, \psi)^{\chi(X)},$$

with the usual convention $0^0 = 1$.