Pushout construction for the Kenzo systems

Jónathan Heras

February 3, 2010

1 Basics on Homological Algebra

The following basic definitions can be found, for instance, in [9].

Definition 1.1. Let R be a ring with a unit element $1 \neq 0$. A *left R-module* M is an additive Abelian group together with a map $p : R \times M \to M$, denoted by $p(r, m) \equiv rm$, such that for every $r, r' \in R$ and $m, m' \in M$

$$\begin{aligned} (r + r')m &= rm + r'm \\ r(m + m') &= rm + rm' \\ (rr')m &= r(r'm) \\ 1m &= m \end{aligned}$$

A similar definition is given for a *right R-module*. Unless the distinction being necessary, we will talk of an *R-module* M without specifying if it is a right or a left R-module.

For $R = \mathbb{Z}$ (the integer ring), a $\mathbb{Z}$-module M is simply an Abelian group. The map $p : \mathbb{Z} \times M \to M$ is given by

$$p(n, m) = \begin{cases} m + \overset{n}{\cdots} + m & \text{if } n > 0 \\ 0 & \text{if } n = 0 \\ (-m) + \overset{-n}{\cdots} + (-m) & \text{if } n < 0 \end{cases}$$

Definition 1.2. A subset S of an R-module M is a *submodule* if S is closed under addition and for all elements $r \in R$ and $s \in S$, one has $rs \in S$.

One can easily observe that a submodule S of and R-module M is itself an R-module.

Definition 1.3. Let R be a ring and M and N be R-modules. An *R-module morphism* $\alpha : M \to N$ is a function from M to N such that for every $m, m' \in M$ and $r \in R$

$$\begin{aligned} \alpha(m + m') &= \alpha(m) + \alpha(m') \\ \alpha(rm) &= r\alpha(m) \end{aligned}$$

Definition 1.4. Given a ring R, a *chain complex* C_* of R-modules is a pair of sequences $C_* = (C_n, d_n)_{n \in \mathbb{Z}}$ where, for each degree $n \in \mathbb{Z}$, C_n is an R-module, the homogeneous component of degree n of $C = \bigoplus_{n\in\mathbb{Z}} C_n$, and $d_n : C_n \to C_{n-1}$ (*the differential map*) is an R-module morphism (of degree -1) such that $d_{n-1} \circ d_n = 0$ for all n.

The module C_n is called the module of *n-chains*. The image $B_n = \operatorname{Im} d_{n+1} \subseteq C_n$ is the (sub)module of *n-boundaries*. The kernel $Z_n = \operatorname{Ker} d_n \subseteq C_n$ is the (sub)module of *n-cycles*.

Given a chain complex $C_* = (C_n, d_n)_{n\in\mathbb{Z}}$, the identities $d_{n-1} \circ d_n = 0$ are equivalent to the inclusion relations $B_n \subseteq Z_n$: every boundary is a cycle. But the converse in general is not true. Thus the next definition makes sense.

Definition 1.5. Let $C_* = (C_n, d_n)_{n\in\mathbb{Z}}$ be a chain complex of R-modules. For each degree $n \in \mathbb{Z}$, the *n-homology module* of C_* is defined as the quotient module

$$H_n(C_*) = \frac{Z_n}{B_n}$$

Definition 1.6. A *morphism of chain complexes of R-modules* (or a *chain complex morphism*) $f : C_* \to D_*$ between two chain complexes of R-modules $C_* = (C_n, d_{C_n})_{n\in\mathbb{Z}}$ and $D_* = (D_n, d_{D_n})_{n\in\mathbb{Z}}$ is a graded R-module morphism (degree 0) which commutes with the differential map. In other words, f consists of R-module morphisms $f_n : C_n \to D_n$ satisfying $d_{D_n} \circ f_n = f_{n-1} \circ d_{C_n}$ for each n.

It is not difficult to prove that a chain complex morphism $f : C_* \to D_*$ induces an R-module morphism on the corresponding homology modules

$$H_*(f) : H_*(C_*) \longrightarrow H_*(D_*)$$

Definition 1.7. Let $f, g : C_* \to D_*$ be morphisms of chain complexes of R-modules. A *(chain) homotopy* h from f to g, written $h : f \simeq g$, is a set of R-module morphisms $h_n : C_n \to D_{n+1}$ such that $h_{n-1} \circ d_{C_n} + d_{D_{n+1}} \circ h_n = f_n - g_n$ for all n.

Theorem 1.8. [9] Given $f, g : C_* \to D_*$ chain complex morphisms and $h : f \simeq g$ a chain homotopy, then the morphisms induced by f and g on homology are the same:

$$H_n(f) = H_n(g) : H_n(C_*) \longrightarrow H_n(D_*) \quad \text{for all } n \in \mathbb{Z}$$

Definition 1.9. A chain complex morphism $f : C_* \to D_*$ is said to be a *chain equivalence* if there exist a morphism $g : D_* \to C_*$ and homotopies $h_1 : \mathrm{Id}_{C_*} \simeq g \circ f$ and $h_2 : \mathrm{Id}_{D_*} \simeq f \circ g$.

Corollary 1.10. [9] If $f : C_* \to D_*$ is a chain equivalence, the induced map

$$H_n(f) : H_n(C_*) \longrightarrow H_n(D_*)$$

is an isomorphism for each dimension n.

Definition 1.11. A chain complex $C_* = \{C_q, d_q\}_{q\in\mathbb{Z}}$ is *exact at degree q* if *ker d_q = im d_{q+1}*, in other words if $H_q(C_*) = 0$, or if $Z_q(C_*) = B_q(C_*)$: every q-cycle is a q-boundary. The chain complex is *exact* if it is exact in every degree. In the same case, it is frequent also to state the chain complex is *acyclic*.

Proposition 1.12. Let (C_*, d) be a chain complex. If there exists a homotopy operator $h : C_* \to C_{*+1}$ satisfying $id = dh + hd$, then the chain complex (C_*, d) is acyclic.

Definition 1.13. A short exact sequence is a sequence of modules:

$$0 \leftarrow C'' \overset{j}{\leftarrow} C \overset{i}{\leftarrow} C' \leftarrow 0$$

which is exact, that is in this case, the map i is injective, the map j is surjective and $im\ i = ker\ j$.

2 Basics on Simplicial Topology

Simplicial sets were first introduced by Eilenberg and Zilber [5], who called them *semi-simplicial complexes.* They can be used to express some topological properties of spaces by means of combinatorial notions. A good reference for the definitions and results of this section is [11].

Definition 2.1. A *simplicial set* K, is a union $K = \bigcup_{q \geq 0} K^q$, where the K^q are disjoints sets, together with functions:

$$\begin{array}{lll} \partial_i^q : K^q \to K^{q-1}, & q > 0, & i = 0, \dots, q, \\ \eta_i^q : K^q \to K^{q+1}, & q \geq 0, & i = 0, \dots, q, \end{array}$$

subject to the relations:

$$\begin{array}{llcccc} (1) & \partial_i^{q-1}\partial_j^q & = & \partial_{j-1}^{q-1}\partial_i^q & \text{if} & i < j, \\ (2) & \eta_i^{q+1}\eta_j^q & = & \eta_j^{q+1}\eta_{i-1}^q & \text{if} & i > j, \\ (3) & \partial_i^{q+1}\eta_j^q & = & \eta_{j-1}^{q-1}\partial_i^q & \text{if} & i < j, \\ (4) & \partial_i^{q+1}\eta_i^q & = & identity & = & \partial_{i+1}^{q+1}\eta_i^q, \\ (5) & \partial_i^{q+1}\eta_j^q & = & \eta_j^{q-1}\partial_{i-1}^q & \text{if} & i > j+1, \end{array}$$

the ∂_i^q and η_i^q are called *face* and *degeneracy* operators respectively.

The elements of K^q are called *q-simplices.* A simplex x is *degenerate* if $x = \eta_i y$ for some simplex y and degeneracy operator η_i; otherwise x is *non degenerate.* We call an *abstract simplex* a pair consisting of a (possibly iterated) degeneracy operator and a simplex.

Property 2.2. Let K be a simplicial set. Any degenerate n-simplex $x \in K^n$ can be expressed in a unique way as a (possibly) iterated degeneracy of a non-degenerate simplex y in the following way:

$$x = \eta_{j_k} \dots \eta_{j_1} y$$

with $y \in K^r$, $k = n - r > 0$, and $0 \leq j_1 < \cdots < j_k < n$

Another useful example of simplicial set is the *standard m-simplex* $\Delta[m]$.

Definition 2.3. For $m \geq 0$, the *standard m-simplex* $\Delta[m]$ is a simplicial set built as follows. An n-simplex of $\Delta[m]$ is any $(n + 1)$-tuple $(a_0, \dots, a_n)$ of integers such that $0 \leq a_0 \leq \cdots \leq a_n \leq m$, and the face and degeneracy operators are defined as

$$\partial_i(a_0, \dots, a_n) = (a_0, \dots, a_{i-1}, a_{i+1}, \dots, a_n)$$
$$\eta_i(a_0, \dots, a_n) = (a_0, \dots, a_i, a_i, a_{i+1} \dots, a_n)$$

Let K be a simplicial set and $\star \in K_0$ a chosen 0-simplex (called the *base point*). We will also denote by $\star$ the degenerate simplices $\eta_{n-1} \dots \eta_0 \star \in K_n$ for every n.

Definition 2.4. A simplicial set K is said to be *reduced* (or 0-*reduced*) if $K_0 = \{\star\}$, in other words, if K has only one 0-simplex. Given $m \geq 1$, K is m-*reduced* if $K_n = \{\star\}$ for all $n \leq m$.

The categories $\mathcal{S}$ of simplicial sets and $\mathcal{C}$ of chain complexes of $\mathbb{Z}$-modules are closely connected: given a simplicial set K, it is possible to construct, in a very easy way, an associated chain complex.

Definition 2.5. Let K be a simplicial set, we define the *chain complex associated with* K, $C_*(K) = (C_n(K), d_n)_{n\in\mathbb{N}}$, in the following way:

- $C_n(K) = \mathbb{Z}[K_n]$ is the free $\mathbb{Z}$-module generated by K_n. Therefore an n-chain $c \in C_n(K)$ is a combination $c = \sum_{i=1}^m \lambda_i x_i$ with $\lambda_i \in \mathbb{Z}$ and $x_i \in K_n$ for $1 \leq i \leq m$;
- the differential map $d_n : C_n(K) \to C_{n-1}(K)$ is given by

  $$d_n(x) = \sum_{i=0}^{n} (-1)^i \partial_i(x) \text{ for } x \in K_n$$

  and it is extended by linearity to the combinations $c = \sum_{i=1}^m \lambda_i x_i \in C_n(K)$.

Let us remark that if a simplex $x \in K_n$ is degenerate, $x = \eta_j y$ with $0 \leq j < n$ and $y \in K_{n-1}$, then $d_n(x)$ is a sum of degenerate $(n-1)$-simplices:

$$\begin{aligned} d_n(\eta_j y) = \sum_{i=0}^{n} (-1)^i \partial_i \eta_j y &= \sum_{i=0}^{j-1} (-1)^i \eta_{j-1} \partial_i y + (-1)^j y + (-1)^{j+1} y \\ + \sum_{i=j+2}^{n} (-1)^i \eta_j \partial_{i-1} y &= \sum_{i=0}^{j-1} (-1)^i \eta_{j-1} (\partial_i y) + \sum_{i=j+2}^{n} (-1)^i \eta_j (\partial_{i-1} y) \end{aligned}$$

As a consequence, the next definition makes sense.

Definition 2.6. The *normalized (non-degenerate) chain complex associated with* K, $C_*^N(K) = (C_n^N(K), d_n^N)_{n\in\mathbb{N}}$, is given by

- $C_n^N(K) = C_n(K)/\mathbb{Z}[D_n(K)]$, where $D_n(K)$ is the set of degenerate elements of K_n. We can also think of $C_n^N(K)$ as the free $\mathbb{Z}$-module generated by the set of non-degenerate n-simplices of K, denoted by $ND_n(K)$. This means that an n-chain $c \in C_n^N(K)$ is a combination $c = \sum_{i=1}^m \lambda_i x_i$ where $\lambda_i \in \mathbb{Z}$ and x_i is a non-degenerate n-simplex of K for all $1 \leq i \leq m$;
- the differential map $d_n^N : C_n^N(K) \to C_{n-1}^N(K)$ is given by

  $$d_n^N(x) = \sum_{i=0}^{n} (-1)^i \partial_i(x) \quad \text{mod } ND_{n-1}(K) \text{ for } x \in ND_n(K)$$

  We observe that $d_n^N(x)$ is obtained from $d_n(x)$ by canceling the degenerate simplices. The definition is extended by linearity to the combinations of $C_n^N(K)$.

Definition 2.7. Given a simplicial set K, the *n-homology group* of K, $H_n(K)$, is the n-homology group of the chain complex $C_*(K)$:

$$H_n(K) = H_n(C_*(K))$$

Definition 2.8. Given two simplicial sets K and L, the Cartesian product $K \times L$ is a simplicial set with n-simplices

$$(K \times L)_n = K_n \times L_n$$

and if $(x, y) \in K_n \times L_n$, the face and degeneracy operators are defined as

$$\begin{aligned} \partial_i(x, y) &= (\partial_i x, \partial_i y) \quad \text{for } 0 \leq i \leq n \\ \eta_i(x, y) &= (\eta_i x, \eta_i y) \quad \text{for } 0 \leq i \leq n \end{aligned}$$

Definition 2.9. A *(Abelian) simplicial group* G is a simplicial object over the category of (Abelian) groups, in other words, it is a simplicial set where each G_n is an (Abelian) group and the face and degeneracy operators are group morphisms.

An important case of Abelian simplicial groups are the Eilenberg-MacLane spaces.

Definition 2.10. An *Eilenberg-MacLane space* of type (π, n) is a simplicial group K (with base point $e_0 \in K^0$ such that $\pi_n(K) = \pi$ and $\pi_i(K) = 0$ if $i \neq n$. The simplicial group K is called a $K(\pi, n)$ if it is an Eilenberg-MacLane space of type (π, n) an in addition it is minimal.

In order to construct the spaces $K(\pi, n)$'s several methods can be used, although the results are necessarily isomorphic [11]. Let us consider the following one.

Let π be an Abelian group. The simplicial Abelian group $K = K(\pi, 0)$ is given by $K^n = \pi$ for all $n \geq 0$, and with face and degeneracy operators $\partial_i : K^n = \pi \to K^{n-1} = \pi$ and $\eta_i : K^n = \pi \to K^{n+1} = \pi$, $0 \leq i \leq n$, equal to the identity map of the group π.

The space $K(\pi, n)$ can be built recursively by means of the classifying space constructor.

Definition 2.11. Let G be a simplicial Abelian group. The *classifying space* of G, written $B(G)$, is the simplicial Abelian group built as follows. The n-simplices of $B(G)$ are the elements of the Cartesian product

$$B(G)^n = G^{n-1} \times G^{n-2} \times \cdots G^0.$$

In this way $B(G)^0$ is the null group and has only one element that we denote by [ ]. For $n \geq 1$, an element of $B(G)^n$ has the form $[g_{n-1}, \ldots, g_0]$ with $g_i \in G^i$. The face and degeneracy operators are given by

$$\begin{aligned} \eta_0[\,] &= [e_0] \\ \partial_i[g_0] &= [\,], \quad i = 0, 1 \\ \partial_0[g_{n-1}, \ldots, g_0] &= [g_{n-2}, \ldots, g_0] \\ \partial_i[g_{n-1}, \ldots, g_0] &= [\partial_{i-1} g_{n-1}, \ldots, \partial_1 g_{n-i+1}, \partial_0 g_{n-i} + g_{n-i-1}, g_{n-i-2}, \ldots, g_0], \quad 0 < i \leq n \\ \eta_0[g_{n-1}, \ldots, g_0] &= [e_n, g_{n-1}, \ldots, g_0] \\ \eta_i[g_{n-1}, \ldots, g_0] &= [\eta_{i-1} g_{n-1}, \ldots, \eta_0 g_{n-i}, e_{n-i}, g_{n-i-1}, \ldots, g_0], \quad 0 < i \leq n \end{aligned}$$

where e_n denotes the null element of the Abelian group G^n.

We define inductively $B^n(G) = B(B^{n-1}(G))$ for all $n \geq 1$, $B^0(G) = G$.

Theorem 2.12. [11] Let π be an Abelian group and $K(\pi, 0)$ as explained before. Then $B^n(K)$ is a $K(\pi, n)$.

3 Basics on Effective Homology

In this section, we present some definitions (including the notion of object with effective homology) and fundamental results about the effective homology method. More details can be found in [12] and [13].

Definition 3.1. An *effective chain complex* is a *free* chain complex of $\mathbb{Z}$-modules $C_* = (C_n, d_n)_{n\in\mathbb{N}}$ where each group C_n is finitely generated, a provided algorithm returns a (distinguished) $\mathbb{Z}$-basis in each degree n, and each differential map d_n is also given by an algorithm.

If a chain complex $C_* = (C_n, d_n)_{n\in\mathbb{N}}$ is effective, the differential maps $d_n : C_n \to C_{n-1}$ can be expressed as finite integer matrices, and then it is possible to know *everything* about C_*: we can compute the subgroups $\operatorname{Ker} d_n$ and $\operatorname{Im} d_{n+1}$, we can determine whether an n-chain $c \in C_n$ is a cycle or a boundary, and in the last case, we can obtain $z \in C_{n+1}$ such that $c = d_{n+1}(z)$. In particular an elementary algorithm computes its homology groups using, for example, the Smith Normal Form technique (for details, see [8]).

On the other hand, in many situations we must deal with *locally effective* chain complexes. In this case we can have an infinite number of generators for each group C_n, so that no *global* information is available. For example, it is not possible in general to compute the subgroups $\operatorname{Ker} d_n$ and $\operatorname{Im} d_{n+1}$, which can have infinite nature. However, "local"[1] information can be obtained: we can compute, for instance, the boundary of a given element.

More generally, we talk of *locally effective objects* when only "local" computations are possible. For instance, we can consider a locally effective simplicial set; the set of n-simplices is not necessarily of finite type, but we can compute the faces of any specific n-simplex.

The effective homology technique consists in combining locally effective objects with effective chain complexes by means of chain equivalences. In this way, we will be able to compute homology groups of locally effective objects even if we cannot obtain global information about them.

Definition 3.2. A *reduction* ρ (also called *contraction* by other authors) between two chain complexes C_* and D_*, denoted in this memoir by $\rho : C_* \Rrightarrow D_*$, is a triple $\rho = (f, g, h)$

$$\begin{tikzcd} C_* \arrow[loop left, "h"] \arrow[r, bend left, "f"] & D_* \arrow[l, bend left, "g"] \end{tikzcd}$$

where f and g are chain complex morphisms, h is a graded group morphism of degree $+1$, and the following relations are satisfied:

[1]The qualifier "componentwise" would be more appropriate, but a little heavy.

1) $f \circ g = \mathrm{Id}_{D_*}$;

2) $d_C \circ h + h \circ d_C = \mathrm{Id}_{C_*} - g \circ f$;

3) $f \circ h = 0$; $\quad h \circ g = 0$; $\quad h \circ h = 0$.

We observe that this is a particular case of chain equivalence (Definition 1.9), where $h_1 = h : \mathrm{Id}_{C_*} \simeq g \circ f$ and the second homotopy $h_2 : \mathrm{Id}_{D_*} \simeq f \circ g$ is the null map.

These relations express that C_* is the direct sum of D_* and an acyclic chain complex. This decomposition is simply $C_* = \operatorname{Ker} f \oplus \operatorname{Im} g$, with $\operatorname{Im} g \cong D_*$ and $H_*(\operatorname{Ker} f) = 0$. In particular, this implies that the graded homology groups $H_*(C_*)$ and $H_*(D_*)$ are canonically isomorphic.

Very frequently, the *small* chain complex D_* is effective, so that we can compute its homology groups by means of elementary operations with integer matrices. On the other hand, in many situations the *big* chain complex C_* is locally effective and therefore its homology groups cannot directly be determined. However, if we know a reduction from C_* over D_* and D_* is effective, then we are also able to compute the homology groups of C_* by means of those of D_*.

Given a chain complex C_*, a *trivial reduction* $\rho = (f, g, h) : C_* \Rrightarrow C_*$ can be constructed, where f and g are the identity map and $h = 0$.

As we see in the next proposition, the composition of two reductions can be easily constructed.

Proposition 3.3. Let $\rho = (f, g, h) : C_* \Rrightarrow D_*$ and $\rho' = (f', g', h') : D_* \Rrightarrow E_*$ be two reductions. Another reduction $\rho'' = (f'', g'', h'') : C_* \Rrightarrow E_*$ is defined by:

$$\begin{aligned} f'' &= f' \circ f \\ g'' &= g \circ g' \\ h'' &= h + g \circ h' \circ f \end{aligned}$$

Definition 3.4. A *strong chain equivalence* ε between two chain complexes C_* and D_*, denoted by $\varepsilon : C_* \Lleftarrow\!\!\Rrightarrow D_*$, is a triple (B_*, ρ_1, ρ_2) where B_* is a chain complex, and ρ_1 and ρ_2 are reductions from B_* over C_* and D_* respectively:

$$\begin{tikzcd} & B_* \arrow[dl, Rightarrow, "\rho_1"'] \arrow[dr, Rightarrow, "\rho_2"] & \\ C_* & & D_* \end{tikzcd}$$

Details about the different components in the new reductions can be found in [13].

Once we have introduced the notion of equivalence, it is possible to give the definition of *object with effective homology*, which is the fundamental idea of the effective homology technique.

Definition 3.5. An *object with effective homology* X is a quadruple $(X, C_*(X), HC_*, \varepsilon)$ where

- X is a locally effective object;
- $C_*(X)$ is a (locally effective) chain complex canonically associated with X, that allows us to study the homological nature of X;

- HC_* is an effective chain complex;
- ε is an equivalence $\varepsilon : C_*(X) \Lleftarrow\!\!\Rrightarrow HC_*$.

For instance, if K is a simplicial set, then the chain complex canonically associated with K, $C_*(K)$, is described in Definition 2.5. Equivalently, we can also consider the normalized chain complex $C_*^N(K)$ introduced in Definition 2.6. It has been already said that the homology groups of both chain complexes are isomorphic, and in fact there exists a reduction $C_*(K) \Rrightarrow C_*^N(K)$.

Theorem 3.6. Let K be a simplicial set, $C_*(K)$ the chain complex associated with K, and $C_*^N(K)$ the normalized chain complex. Then it is possible to build a reduction $\rho : C_*(K) \Rrightarrow C_*^N(K)$.

In this way, K is a simplicial set with effective homology if an equivalence between $C_*(K)$ or $C_*^N(K)$ and an effective chain complex is known. Clearly, using Theorem 3.6 and the composition of reductions and equivalences, from and equivalence $\varepsilon : C_*(K) \Lleftarrow\!\!\Rrightarrow HC_*$ we can determine $\varepsilon' : C_*^N(K) \Lleftarrow\!\!\Rrightarrow HC_*$ and reciprocally an equivalence $\varepsilon' : C_*^N(K) \Lleftarrow\!\!\Rrightarrow HC_*$ allows us to construct $\varepsilon : C_*(K) \Lleftarrow\!\!\Rrightarrow HC_*$.

It is clear that if X is an object with effective homology, then the homology groups of X (which are those of the associated chain complex $C_*(X)$) are isomorphic to the homology groups of the effective chain complex HC_*, that can easily be computed using some elementary operations. But it is important to understand that in general the HC_* component of an object with effective homology is not made of the homology groups of X; this component HC_* is a *free* $\mathbb{Z}$-chain complex of finite type, in general with a non-null differential.

The main problem now is the following: given a chain complex $C_* = (C_n, d_n)_{n \in \mathbb{N}}$, is it possible to determine its effective homology? We must distinguish three cases.

- First of all, if a chain complex C_* is by chance effective, then we can choose the trivial effective homology: ε is the equivalence $C_* \Lleftarrow C_* \Rrightarrow C_*$, where the two components ρ_1 and ρ_2 are both the trivial reduction on C_*.
- In some cases, some theoretical results are available providing an equivalence between some chain complex C_* and an *effective* chain complex. Typically, the Eilenberg-MacLane space $K(\mathbb{Z}, 1)$ has the homotopy type of the circle S^1 and a reduction $C_*(K(\mathbb{Z}, 1)) \Rrightarrow C_*(S^1)$ can be built.
- The most important case: let $X_1, \ldots, X_n$ be objects with effective homology and Φ a constructor that produces a new space $X = \Phi(X_1, \ldots, X_n)$ (for example, the Cartesian product of two simplicial sets, the classifying space of a simplicial group, etc). In *natural* "reasonable" situations, there exists an effective homology version of Φ that allows us to deduce a version with effective homology of X, the result of the construction, from versions with effective homology of the arguments $X_1, \ldots, X_n$.

For instance, given two simplicial sets K and L with effective homology, then the Cartesian product $K \times L$ is an object with effective homology too, and this is also valid for twisted Cartesian products. We will see in Section **??** how this effective homology is obtained.

The next two theorems will be useful when obtaining the effective homology version of some topological constructors. The main idea is that given a reduction, if we *perturb* one of the complexes then it is possible to perturb the other one so that we obtain a new reduction between the *perturbed* complexes. The first theorem (the Trivial Perturbation Lemma) is very easy, but it can be useful. The Basic Perturbation Lemma is not trivial at all. It was discovered by Shih Weishu [14], although the abstract modern form was given by Ronnie Brown [1].

Definition 3.7. Let $C_* = (C_n, d_n)_{n\in\mathbb{N}}$ be a chain complex. A *perturbation* δ of the differential d is a collection of group morphisms $\delta = \{\delta_n : C_n \to C_{n-1}\}_{n\in\mathbb{N}}$ such that the sum $d + \delta$ is also a differential, that is to say, $(d+\delta)\circ(d+\delta) = 0$.

The perturbation δ produces a new chain complex $C'_* = (C_n, d_n + \delta_n)_{n\in\mathbb{N}}$; it is the *perturbed* chain complex.

Theorem 3.8 (Trivial Perturbation Lemma, TPL)**.** Let $C_* = (C_n, d_{C_n})_{n\in\mathbb{N}}$ and $D_* = (D_n, d_{D_n})_{n\in\mathbb{N}}$ be two chain complexes, $\rho = (f, g, h) : C_* \Rrightarrow D_*$ a reduction, and δ_D a perturbation of d_D. Then a new reduction $\rho' = (f', g', h') : C'_* \Rrightarrow D'_*$ can be constructed where:

1) C'_* is the chain complex obtained from C_* by replacing the old differential d_C by the perturbed differential $(d_C + g\circ\delta_D\circ f)$;

2) the new chain complex D'_* is obtained from the chain complex D_* only by replacing the old differential d_D by $(d_D + \delta_D)$;

3) $f' = f$;

4) $g' = g$;

5) $h' = h$.

The perturbation δ_D of the *small* chain complex D_* is naturally transferred (using the reduction ρ) to the *big* chain complex C_*, obtaining in this way a new reduction ρ' (which in fact has the same components as ρ) between the perturbed chain complexes. On the other hand, if we consider a perturbation d_C of the top chain complex C_*, in general it is not possible to perturb the small chain complex D_* so that there exists a reduction between the perturbed chain complexes. As we will see, we need an additional hypothesis.

Theorem 3.9 (Basic Perturbation Lemma, BPL)**.** [1] Let us consider a reduction $\rho = (f, g, h) : C_* \Rrightarrow D_*$ between two chain complexes $C_* = (C_n, d_{C_n})_{n\in\mathbb{N}}$ and $D_* = (D_n, d_{D_n})_{n\in\mathbb{N}}$, and δ_C a perturbation of d_C. Furthermore, the composite function $h\circ\delta_C$ is assumed *locally nilpotent*, in other words, given $x \in C_*$ there exists $m \in \mathbb{N}$ such that $(h\circ\delta_C)^m(x) = 0$. Then a new reduction $\rho' = (f', g', h') : C'_* \Rrightarrow D'_*$ can be constructed where:

1) C'_* is the chain complex obtained from the chain complex C_* by replacing the old differential d_C by $(d_C + \delta_C)$;

2) the new chain complex D'_* is obtained from D_* by replacing the old differential d_D by $(d_D + \delta_D)$, with $\delta_D = f\circ\delta_C\circ\phi\circ g = f\circ\psi\circ\delta_C\circ g$;

3) $f' = f \circ \psi = f \circ (\mathrm{Id}_{C_*} - \delta_C \circ \phi \circ h)$;

4) $g' = \phi \circ g$;

5) $h' = \phi \circ h = h \circ \psi$;

with the operators ϕ and ψ defined by

$$\phi = \sum_{i=0}^{\infty} (-1)^i (h \circ \delta_C)^i$$
$$\psi = \sum_{i=0}^{\infty} (-1)^i (\delta_C \circ h)^i = \mathrm{Id}_{C_*} - \delta_C \circ \phi \circ h,$$

the convergence of these series being ensured by the locally nilpotency of the compositions $h \circ \delta_C$ and $\delta_C \circ h$.

The *cone constructor* is important in homological algebra, and we present here the most elementary properties, a complete study can be found in [13]. The following definitions and theorems will be used in order to generate the effective homology of the pushout as we will see in Section .

Definition 3.10. Let C_* and D_* be two chain complexes and $\phi : C_* \leftarrow D_*$ be a chain-complex morphism. Then the cone of ϕ denoted by $Cone(\phi)$ is the chain complex $Cone(\phi) = A_*$ defined as follows. First $A_n := C_{n+1} \oplus D_n$; then the boundary operator is given by the matrix:

$$d_{A_*} := \begin{bmatrix} d_{C_*} & \phi \\ 0 & -d_{D_*} \end{bmatrix}$$

Theorem 3.11 (Cone Reduction Theorem)**.** Let $\rho = (f, g, h) : C_* \Rrightarrow D_*$ and $\rho' = (f', g', h') : C'_* \Rrightarrow D'_*$ be two reductions and $\phi : C_* \leftarrow C'_*$ a chain complex morphism. THen these data define a canonical reduction:

$$\rho'' = (f'', g'', h'') : Cone(\phi) \Rrightarrow Cone(f \phi g)$$

Theorem 3.12 (Cone Equivalence Theorem)**.** Let $\phi : C_{*,EH} \leftarrow C'_{*,EH}$ be a chain complex morphism between two chain complexes with effective homology. Then a general algorithm computes a version with effective homology $Cone(\phi)_{EH}$ of the cone.

$$\begin{tikzcd}
& \hat{C}_* \arrow[loop left, "lh"] \arrow[loop right, "rh"] \arrow[dl, shift left, "lf"] \arrow[dr, shift left, "rf"] & & & \hat{C'}_* \arrow[llll, "\hat{\phi}"'] \arrow[loop left, "lh'"] \arrow[loop right, "rh'"] \arrow[dl, shift left, "lf'"] \arrow[dr, shift left, "rf'"] & \\
C_* \arrow[ur, shift left, "lg"] & & EC_* \arrow[ul, shift left, "rg"] & C'_* \arrow[ur, shift left, "lg'"] \arrow[lll, bend left, "\phi"] & & EC'_* \arrow[ul, shift left, "rg'"] \arrow[lll, bend left, "E\phi"]
\end{tikzcd}$$

Definition 3.13. An *effective short exact sequence* of chain complexes is a diagram:

$$\begin{tikzcd} 0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

where i and j are chain complexes morphisms, ρ (retraction) and σ (section) are graded module morphisms satisfying:

- $\rho i = id_{C_*}$;
- $i\rho + \sigma j = id_{B_*}$;
- $j\sigma = id_{A_*}$.

It is an exact sequence in both directions, but to the left it is an exact sequence of chain complexes, and to the right it is only an exact sequence of graded modules.

Theorem 3.14 (SES Theorems)**.** Let

$$\begin{tikzcd} 0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

be an effective short exact sequence of chain-complexes. Then three general algorithms are available:

$$\begin{array}{c} SES_1 : (B_{*,EH}, C_{*,EH}) \mapsto A_{*,EH} \\ SES_2 : (A_{*,EH}, C_{*,EH}) \mapsto B_{*,EH} \\ SES_3 : (A_{*,EH}, B_{*,EH}) \mapsto C_{*,EH} \end{array}$$

Lemma 3.15. Let

$$\begin{tikzcd} 0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

be an effective short exact sequence of chain-complexes. Then the effective exact sequence produces a reduction $Cone(i) \Rrightarrow A_*$.

Lemma 3.16. Let

$$\begin{tikzcd} 0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

be an effective short exact sequence of chain-complexes. Then the effective exact sequence generates a connection chain complex morphism $\chi : A_* \to C_*^{[1]}$. The "exponent" [1] explains the *suspension functor* is applied to the chain complex C_*: the degree of an element is increased by 1 and the differential is replaced by the opposite. Besides, B_* is canonically isomorphic to $Cone(\chi)$.

4 The Kenzo program

Kenzo is a 16000 lines program written in Common Lisp [6], devoted to Symbolic Computation in Algebraic Topology. It was developed by Francis Sergeraert and some co-workers, and is www-available (see [4] for documentation and details). It works with the main mathematical structures used in Simplicial Algebraic Topology, [7], (chain complexes, differential graded algebras, simplicial sets, morphisms between these objects, reductions and so on) and has obtained some results (for example, homology groups of iterated loop spaces of a loop space modified by a cell attachment) which have not been confirmed nor refuted by any other means.

The fundamental idea of the Kenzo system is the notion of *object with effective homology* combined with *functional programming*. By using these two tools, not only *known* algorithms were implemented but also new methods were developed to *transform* the main "tools" of Algebraic Topology, mainly the spectral sequences, not at all *algorithmic* in the traditional organization, into actual *computing* methods.

Making use of the existing tools in Common Lisp for inheritance between classes (in CLOS, the Common Lisp Object System), the main mathematical structures of Simplicial Algebraic Topology and relations among them are represented in Kenzo. For instance, the following class definition corresponds to the simplest algebraic structure implemented in Kenzo, the Chain complexes:

```
(DEFCLASS CHAIN-COMPLEX ()
    ((cmpr  :type cmprf :initarg :cmpr  :reader cmpr1)
     (basis :type basis :initarg :basis :reader basis1)
     ;; BaSe GeNerator
     (bsgn :type gnrt :initarg :bsgn :reader bsgn)
     ;; DiFFeRential
     (dffr :type morphism :initarg :dffr :reader dffr1)
     ;; GRound MoDule
     (grmd :type chain-complex :initarg :grmd :reader grmd)
     ;; EFfective HoMology
     (efhm :type homotopy-equivalence :initarg :efhm :reader efhm)
     ;; IDentification NuMber
     (idnm :type fixnum :initform (incf *idnm-counter*) :reader idnm)
     ;; ORiGiN
     (orgn :type list :initarg :orgn :reader orgn)))
```

The relevant slots are `cmpr`, a function coding the equality between the generators; `basis`, the function defining the distinguished ordered basis of each group of n-chains, or the keyword `:locally-effective` if the chain complex is not effective; `dffr`, the differential morphism, which is an instance of the class `MORPHISM`; `efhm`, which stores information about the effective homology of the chain complex; and `orgn`, used to keep record of information about the object.

The class `CHAIN-COMPLEX` is extended by inheritance with new slots, obtaining more elaborate structures. For instance, extending it with an `aprd` (algebra product) slot, we obtain the `ALGEBRA` class. Multiple inheritance is also available; for example, the class `SIMPLICIAL-GROUP` is obtained by inheritance from the classes `KAN` and `HOPF-ALGEBRA`.

It is worth emphasizing here that simplicial sets have also been implemented as a subclass of `CHAIN-COMPLEX`. To be precise, the class `SIMPLICIAL-SET` inherits from the class `COALGEBRA`, which is a direct subclass of `CHAIN-COMPLEX`, with a slot `cprd` (the coproduct). The class `SIMPLICIAL-SET` has then one slot of its own: `face`, a Lisp function computing any face of a simplex of the simplicial set. The basis is in this case (when working with effective objects) the list of non-degenerate simplices, and the differential map of the associated chain complex is given by the alternate sum of the faces, where the degenerate simplices are canceled.

Let us show a simple example to illustrate which is possible with the Kenzo program. The homology group $H_5\Omega^3 Moore(\mathbb{Z}_2, 4)$ is "in principle" reachable thanks to old methods, see [2], but experience shows even the most skilful topologist meet some difficulties to determine it, see [12]. With the Kenzo program, you construct the Moore space in the following way:

```
> (setf m4 (moore 2 4)) ✠
[K1 Simplicial-Set]
```

A Kenzo display must be read as follows. The initial `>` is the Lisp prompt of this Common Lisp implementation. The user types out a Lisp statement, here `(setf m4 (moore 2 4))` and the maltese cross ✠ (in fact not visible on the user screen) marks in this text the end of the Lisp statement, just to help the reader: the right number of closing parentheses is reached. The Return key then asks Lisp to *evaluate* the Lisp statement. Here the Moore space $Moore(\mathbb{Z}_2, 4)$ is constructed by the Kenzo function `moore`, taking account of the arguments 2 and 4, and this Moore space is *assigned* to the Lisp symbol `m4` for later use. Also evaluating a Lisp statement *returns* an object, the result of the evaluation, in this case the Lisp object implementing the Moore space, displayed as `[K1 Simplicial-Set]`, that is, the Kenzo object #1, a `Simplicial-Set`. The internal structure of this object, made of a rich set of data, in particular many functional components, is not displayed.

It is then possible to construct the third loop space of this Moore space $\Omega^3 Moore(\mathbb{Z}_2, 4)$, a simplicial *group*.

```
> (setf o3m4 (loop-space m4 3)) ✠
[K30 Simplicial-Group]
```

The combinatorial version of the loop space is *highly* infinite: it is a combinatorial version of the space of *continuos* maps $S^3 \to Moore(\mathbb{Z}_2, 4)$, but functionally codes as a small set of functions in a `Simplicial-Group` object. This object is *locally effective* and no global information is available. For instance if we try to obtain the list of non-degenerate simplices in dimension 3, we obtain an error.

```
> (basis o3m4 3) ✠
Error: The obtect [K30 Simplicial-Group] is locally-effective
```

The key point to compute the homology groups of `o3m4`, it is an equivalence between the *locally effective* chain complex `K30`= $\Omega^3 Moore(\mathbb{Z}_2, 4)$ and an effective chain complex which is not detailed. We can ask for the $\underline{ef}$fective $\underline{h}$o$\underline{m}$ology of `o3m4`.

```
> (efhm o3m4) ✠
[K404 Homotopy-Equivalence K30 <= K394 => K390]
```

So, that the homology groups of $\Omega^3 Moore(\mathbb{Z}_2, 4)$ are computable through the *effective* equivalence object `K390`:

```
> (homology o3m4 5) ✠
Homology in dimension 5:
Component Z/Z2
Component Z/Z2
Component Z/Z2
Component Z/Z2
Component Z/Z2
---done---
```

To be interpreted as stating $H_5\Omega^3 Moore(\mathbb{Z}_2, 4) = \mathbb{Z}_2^5$. In this way, Kenzo computes the homology groups of *complicated* spaces by means of the effective homology method.

5 Pushout preliminaries

The following definitions can be found, for instance, in [10, 3].

Definition 5.1 (Pushout)**.** Consider two morphisms $f : X \to Y$, $g : X \to Z$ in a category $\mathcal{C}$. A *pushout* of (f, g) is a triple (P, f', g') where

1. P is an object of $\mathcal{C}$,

2. $f' : Y \to P$, $g' : Z \to P$ are morphism of $\mathcal{C}$ such that $f \circ g' = g \circ f'$,

and for every other triple (Q, f'', g'') where

1. Q is an object of $\mathcal{C}$,

2. $f'' : Y \to Q$, $g'' : Z \to Q$ are morphism of $\mathcal{C}$ such that $f \circ g'' = g \circ f''$,

there exists a unique morphism $p : P \to Q$ such that $f'' = p \circ f'$ and $g'' = p \circ g'$ (see the following diagram)

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \arrow[ddr, "f''"] & \\
Z \arrow[r, "g'"] \arrow[drr, "g''"'] & P \arrow[dr, dashed, "p"] & \\
& & Q
\end{tikzcd}$$

Definition 5.2 (Homotopy Pushout)**.** A homotopy commutative diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \\
Z \arrow[r, "g'"] & P
\end{tikzcd}$$

equipped with $H : f' \circ f \sim g' \circ g$, is called a *homotopy pushout* when for any commutative diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f''"] \\
Z \arrow[r, "g''"] & Q
\end{tikzcd}$$

equipped with $G : f'' \circ f \sim g'' \circ g$, the following properties hold:

1. there exists a map $p : P \to Q$ and homotopies $K : f'' \sim p \circ f'$ and $L : p \circ g' \sim g''$ such that the whole diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \arrow[ddr, "f''"] & \\
Z \arrow[r, "g'"] \arrow[drr, "g''"'] & P \arrow[dr, dashed, "p"] & \\
 & & Q
\end{tikzcd}$$

   with all maps and homotopies above is homotopy commutative

2. if there exists another map $p' : P \to Q$ and homotopies $K' : f'' \sim p \circ f'$ and $L' : p \circ g' \sim g''$ such that the diagram

$$\begin{tikzcd}
X \arrow[r, "f"] \arrow[d, "g"] & Y \arrow[d, "f'"] \arrow[ddr, "f''"] & \\
Z \arrow[r, "g'"] \arrow[drr, "g''"'] & P \arrow[dr, dashed, "p'"] & \\
 & & Q
\end{tikzcd}$$

   is homotopy commutative , then there exists a homotopy $M : p \sim p'$ such that the whole diagram with all maps and homotopies above is homotopy commutative.

There is a "standard" construction of the homotopy pushout of any two maps $f : X \to Y$, $g : X \to Z$ as:

$$P_{(f,g)} \cong (Y \amalg X \times I \amalg Z)/\sim$$

where the equivalence relation $\sim$ is defined as follows. For every $x \in X$, $(x \times 0)$ is identified to $f(x) \in Y$ and $(x \times 1)$ is identified to $g(x) \in Z$.

Let X, Y and Z be 1-reduced simplicial sets with effective homology such that three equivalences are given:

$$\begin{tikzcd}
 & DX_* \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & & & DY_* \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & & & DZ_* \arrow[dl, Rightarrow] \arrow[dr, Rightarrow] & \\
C_*(X) & & HX_* & C_*(Y) & & HY_* & C_*(Z) & & HZ_*
\end{tikzcd}$$

where HX_*, HY_* and HZ_* are effective chain complexes; and let $f : X \to Y$, $g : X \to Z$ simplicial morphism. Our goal in this section is the computation of the effective homology of the simplicial set $P_{(f,g)}$. This effective homology will be obtained from two short exact sequence and the Theorem 3.14.

Let us consider the short exact sequence

$$\begin{tikzcd} 0 & M \arrow[l] \arrow[r, shift left, "\sigma"] & C(X \times I) \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C(X \times \{0\}) \oplus C(X \times \{1\}) \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$$

where I is the unit interval $[0,1]$ and M is the chain complex coming from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled; and the morphism i, j, σ and ρ are defined as follows:

$$\begin{array}{llcll}
i: & C(X \times \{0\}) \oplus C(X \times \{1\}) & \rightarrow & C(X \times I) & \\
 & x \times \{0\} & \mapsto & x \times \{0\} & \\
 & x \times \{1\} & \mapsto & x \times \{1\} & \\
 & & & & \\
j: & C(X \times I) & \rightarrow & M & \\
 & x \times \{a\} & \mapsto & x \times \{a\} & \text{if } a \neq 0,1 \\
 & x \times \{0\} & \mapsto & nil & \\
 & x \times \{1\} & \mapsto & nil & \\
 & & & & \\
\sigma: & M & \rightarrow & C(X \times I) & \\
 & x \times \{a\} & \mapsto & x \times \{a\} & \\
 & & & & \\
\rho: & C(X \times I) & \rightarrow & C(X \times \{0\}) \oplus C(X \times \{1\}) & \\
 & x \times \{a\} & \mapsto & nil & \text{if } a \neq 0,1 \\
 & x \times \{0\} & \mapsto & x \times \{0\} & \\
 & x \times \{1\} & \mapsto & x \times \{1\} &
\end{array}$$

Besides, three algorithms are available.

Algorithm 1.
Input: two simplicial sets X and Y with effective homology.
Output: an equivalence $C_*(X \times Y) \Lleftarrow DC_*(X \times Y) \Rrightarrow HC_*(X \times Y)$

Algorithm 2.
Input: two simplicial sets X and Y with effective homology.
Output: an equivalence $C_*(X \oplus Y) \Lleftarrow DC_*(X \oplus Y) \Rrightarrow HC_*(X \oplus Y)$

Algorithm 3.
Input: a morphism i between two chain complexes C_*X and C_*Y with effective homology.

$$\begin{tikzcd}
 & DC_*X \arrow[loop above, "lhx"] \arrow[loop above, "rhx"] \arrow[dl, shift left, "lfx"] \arrow[dr, shift left, "rfx"] & & & DC_*Y \arrow[loop above, "lhy"] \arrow[loop above, "rhy"] \arrow[lll, "\hat{i}"] \arrow[dl, shift left, "lfy"] \arrow[dr, shift left, "rfy"] & \\
C_*X \arrow[ur, shift left, "lgx"] & & HC_*X \arrow[ul, shift left, "rgx"] & C_*Y \arrow[ur, shift left, "lgy"] \arrow[lll, bend left, "i"] & & HC_*Y \arrow[ul, shift left, "rgy"] \arrow[lll, bend left, "Hi"]
\end{tikzcd}$$

Output: an equivalence $Cone(i) \Lleftarrow Cone(\hat{i}) \Rrightarrow Cone(Hi)$, where $\hat{i} := (lgx)i(lfy)$ and $Hi := (rfx)\hat{i}(rgy)$.

From the SES_1 Theorem (Theorem 3.14) we can consider the sequence:

$$M \Lleftarrow Cone(i) \Lleftarrow Cone(\hat{i}) \Rrightarrow Cone(Hi)$$

defined as follows.

From algorithms 1 and 2 two equivalences can be dealt, $C_*(X \times I) \Lleftarrow DC(X \times I)_* \Rrightarrow HC(X \times I)_*$ and $C_*(X \times \{0\} \oplus X \times \{1\}) \Lleftarrow DC(X \times \{0\} \oplus X \times \{1\})_* \Rrightarrow HC(X \times \{0\} \oplus X \times \{1\})_*$, and the morphism $i : C_*(X \times I) \leftarrow C_*(X \times \{0\} \oplus X \times \{1\})$. In order to simplify the notation, $C_*(X \times I)$ and $C_*(X \times \{0\} \oplus X \times \{1\})$ are called CI and CO respectively.

$$\begin{tikzcd}
 & DCI_* \arrow[loop above, "lhi"] \arrow[loop above, "rhi"'] \arrow[dl, shift left, "lfi"] \arrow[dr, shift left, "rfi"] & & & DCO_* \arrow[loop above, "lho"] \arrow[loop above, "rho"'] \arrow[llll, "\hat{i}"'] \arrow[dl, shift left, "lfo"] \arrow[dr, shift left, "rfo"] & \\
CI_* \arrow[ur, shift left, "lgi"] & & HCI_* \arrow[ul, shift left, "rgi"] & CO_* \arrow[ur, shift left, "lgo"] \arrow[lll, bend left, "i"] & & HCO_* \arrow[ul, shift left, "rgo"] \arrow[lll, bend left, "Hi"]
\end{tikzcd}$$

The morphism i naturally induces "parallel" morphism $\hat{i} := (lgi)i(lfo) : DCI_* \leftarrow DCO_*$ and then $Hi := (rfi)(lgi)i(lfo)(rgo) : ECI_* \leftarrow ECO_*$.

Applying the Cone Equivalence Theorem (Theorem 3.12), the next equivalence is defined (that is algorithm 3):

$$\begin{tikzcd}
 & Cone(\widehat{i})_* \arrow[loop above, "lh_i"] \arrow[loop above, "rh_i"'] \arrow[dl, shift left, "lf_i"] \arrow[dr, shift left, "rf_i"] & \\
Cone(i)_* \arrow[ur, shift left, "lg_i"] & & Cone(Hi)_* \arrow[ul, shift left, "rg_i"]
\end{tikzcd}$$

where

$$d_{Cone(i)_*} = \begin{bmatrix} d_{CI_*} & i \\ 0 & d_{CO_*} \end{bmatrix}, \; d_{Cone(\hat{i})_*} = \begin{bmatrix} d_{DCI_*} & \widehat{i} \\ 0 & d_{DCO_*} \end{bmatrix}, \; d_{Cone(Hi)_*} = \begin{bmatrix} d_{HCI_*} & Hi \\ 0 & d_{HCO_*} \end{bmatrix}$$

$$lf_i = \begin{bmatrix} lfi & 0 \\ 0 & lfo \end{bmatrix}, \; lg_i = \begin{bmatrix} lgi & 0 \\ 0 & lgo \end{bmatrix}, \; lh_i = \begin{bmatrix} lhi & 0 \\ 0 & -lho \end{bmatrix},$$

$$rf_i = \begin{bmatrix} rfi & (rfi)\hat{i}(rho) \\ 0 & rfo \end{bmatrix}, \; rg_i = \begin{bmatrix} rgi & -(rhi)\hat{i}(rgo) \\ 0 & rgo \end{bmatrix}, \; rh_i = \begin{bmatrix} rhi & (rhi)\hat{i}(rho) \\ 0 & -rho \end{bmatrix}.$$

Now, applying the SES_1 Theorem (Theorem 3.14) we have the sequence $M \Lleftarrow Cone(i) \Lleftarrow Cone(\hat{i})$

$$\begin{tikzcd}
 & & Cone(\widehat{i})_* \arrow[loop above, "lh_i"] \arrow[loop above, "rh_i"] \arrow[dl, shift left, "lf_i"] \arrow[dr, shift left, "rf_i"] & \\
 & Cone(i)_* \arrow[loop above, "hm"] \arrow[ur, shift left, "lg_i"] \arrow[dl, shift left, "fm"] & & Cone(Hi)_* \arrow[ul, shift left, "rg_i"] \\
M \arrow[ur, shift left, "gm"] & & &
\end{tikzcd}$$

where the values for $lf_i, lg_i, lh_i, rf_i, rg_i, rh_i$ where described previously and $fm = j$, $gm = \sigma - \rho d_{CI}\sigma$ and $h1 = \rho$.

Algorithm 4.
Input: the short exact sequence $\begin{tikzcd} 0 & A_* \arrow[l, "0"'] \arrow[r, shift left, "\sigma"] & B_* \arrow[l, shift left, "j"] \arrow[r, shift left, "\rho"] & C_* \arrow[l, shift left, "i"] & 0 \arrow[l] \end{tikzcd}$ where A, B and C are simplicial sets with effective homology and i, j, σ and ρ are morphism.
Output: the equivalence $A \Lleftarrow Cone(i)$.

Applying Proposition 3.3 to the reductions $M \Lleftarrow Cone(i)$ and $Cone(i) \Lleftarrow$ the reduction $M \Lleftarrow Cone(\hat{i})$ is obtained. Then, the following homotopy equivalence is defined.

$$M \Lleftarrow Cone(\widehat{i}) \Rrightarrow Cone(Ei)$$

$$\begin{tikzcd}
 & Cone(\widehat{i})_* \arrow[loop above, "lh'_i"] \arrow[loop above, "rh_i"] \arrow[dl, shift left, "lf'_i"] \arrow[dr, shift left, "rf_i"] & \\
M_* \arrow[ur, shift left, "lg'_i"] & & Cone(Hi)_* \arrow[ul, shift left, "rg_i"]
\end{tikzcd}$$

where $lf'_i = fm \circ lf_i$, $lg'_i = lg_i \circ gm$ and $lh'_i = lh_i + lg_i \circ hm \circ lf_i$.

Algorithm 5.
Input: a simplicial set X.
Output: an equivalence $M \Lleftarrow DM \Rrightarrow HM$, where M is the chain complex coming from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled.

Let us consider, the short exact sequence:

$$\begin{tikzcd} 0 & M \arrow[l] \arrow[r, shift left, "\sigma 2"] & CP \arrow[l, shift left, "j2"] \arrow[r, shift left, "\rho 2"] & CY \oplus CZ \arrow[l, shift left, "i2"] & 0 \arrow[l] \end{tikzcd}$$

where CP is the chain complex coming from the pushout $P_{(f,g)}$ with $f : X \to Y$ and $g : X \to Z$.

In this case the functions $i2, j2, \sigma 2$ and $\rho 2$ are defined as follows:

$$\begin{array}{rccll}
i2: & CY \oplus CZ & \rightarrow & CP & \\
 & y & \mapsto & y & \text{if } y \in CY \\
 & z & \mapsto & z & \text{if } z \in CZ
\end{array}$$

$$\begin{array}{rccll}
j2: & CP & \rightarrow & M & \\
 & x \times \{a\} & \mapsto & x \times \{a\} & \text{if } x \in X \text{ and } a \in I \\
 & y & \mapsto & nil & \text{if } y \in Y \\
 & z & \mapsto & nil & \text{if } z \in Z
\end{array}$$

$$\begin{array}{rccl}
\sigma 2: & M & \rightarrow & CP \\
 & x \times \{a\} & \mapsto & x \times \{a\}
\end{array}$$

$$\begin{array}{rccll}
\rho 2: & CP & \rightarrow & CY \oplus CZ & \\
 & x \times \{a\} & \mapsto & nil & \text{if } x \in X \text{ and } a \in I \\
 & y & \mapsto & y & \text{if } y \in Y \\
 & z & \mapsto & z & \text{if } z \in Z
\end{array}$$

Applying SES_2 Theorem (Theorem 3.14) the above effective short exact sequence generates a connection chain complex morphism: $\chi : M_* \rightarrow (CY \oplus CZ)_*^{[1]}$. The "exponent" [1] explains the suspension functor is applied to the chain-complex $(CY \oplus CZ)_*$: the degree of an element is increased by 1 and the differential is replaced by the opposite. The connection morphism is defined as the composition $\chi = \rho d_{CP} \sigma$.

Algorithm 6.
Input: a chain complex X with effective homology.
Output: an equivalence $X^{[1]} \Lleftarrow DX^{[1]} \Rrightarrow HX^{[1]}$, where $X^{[1]}$ is the chain complex coming from applying the suspension functor to X.

From the two equivalences $M \Lleftarrow Cone(\hat{i}) \Rrightarrow Cone(Hi)$, $C_*(Y \oplus Z) \Lleftarrow DC_*(Y \oplus Z) \Rrightarrow HC_*(Y \oplus Z)$ and the chain complex morphism χ that induces two "parallel" morphisms $\widehat{\chi} = (lgyz)\chi(lf'_i)$ and $H\chi = (rfyz)(lgyz)\chi(lf'_i)(rg_i)$:

$$\begin{tikzcd}
 & D(CY \oplus CZ)_* \arrow[loop above, "lhyz"] \arrow[loop above, "rhyz"'] \arrow[dl, shift left, "lfyz"] \arrow[dr, shift left, "rfyz"] & & & Cone(\widehat{i})_* \arrow[loop above, "lh'_i"] \arrow[loop above, "rh_i"'] \arrow[lll, "\hat{\chi}"] \arrow[dl, shift left, "lg'_i"] \arrow[dr, shift left, "rf_i"] & \\
(CY \oplus CZ)_* \arrow[ur, shift left, "lgyz"] & & H(CY \oplus CZ)_* \arrow[ul, shift left, "rgyz"] & M_* \arrow[ur, shift left, "lf'_i"] \arrow[lll, bend left, "\chi"] & & Cone(Hi)_* \arrow[ul, shift left, "rg_i"] \arrow[lll, bend left, "H\chi"]
\end{tikzcd}$$

The fact that CP is canonically isomorphic to $Cone(\chi)$ is stated in SES_2 Theorem, then the following equivalence is defined:

$$\begin{tikzcd}
 & Cone(\widehat{\chi})_* \arrow[loop above, "lh_{\chi}"] \arrow[loop above, "rh_{\chi}"'] \arrow[dl, shift left, "lf_{\chi}"] \arrow[dr, shift left, "rf_{\chi}"] & \\
CP \cong Cone(\chi)_* \arrow[ur, shift left, "lg_{\chi}"] & & Cone(H\chi)_* \arrow[ul, shift left, "rg_{\chi}"]
\end{tikzcd}$$

where:

$$d_{Cone(\chi)_*} = \begin{bmatrix} d_{(CY\oplus CZ)_*} & \chi \\ 0 & d_{M_*} \end{bmatrix}, \; d_{Cone(\hat{\chi})_*} = \begin{bmatrix} d_{\widehat{(CY\oplus CZ)}_*} & \widehat{\chi} \\ 0 & d_{Cone\widehat{i}_*} \end{bmatrix},$$

$$d_{Cone(H\chi)_*} = \begin{bmatrix} d_{H(CY\oplus CZ)_*} & H\chi \\ 0 & d_{Cone(Hi)_*} \end{bmatrix}, \; lf_{\chi} = \begin{bmatrix} lfyz & 0 \\ 0 & lf'_i \end{bmatrix}, \; lg_{\chi} = \begin{bmatrix} lgyz & 0 \\ 0 & lg'_i \end{bmatrix},$$

$$lh_{\chi} = \begin{bmatrix} lhyz & 0 \\ 0 & -lh'_i \end{bmatrix}, \; rf_{\chi} = \begin{bmatrix} rfyz & (rfyz)\hat{\chi}(rh_i) \\ 0 & rf_i \end{bmatrix}$$

$$rg_{\chi} = \begin{bmatrix} rgi & -(rhyz)\hat{\chi}(rg_i) \\ 0 & rg_i \end{bmatrix}, \; rh_{\chi} = \begin{bmatrix} rhyz & (rhyz)\hat{\chi}(rh_i) \\ 0 & -rh_i \end{bmatrix}$$

Algorithm 7.
Input: two morphism $f : X \to Y$ and $g : X \to Z$ where X, Y and Z are 1-reduced simplicial sets with effective homology.
Output: an equivalence $CP \Lleftarrow DP \Rrightarrow HP$, where CP is the chain complex coming from the pushout of f and g.

The pushout is a generic construction that can be particularized to different constructions, such as wedge and join.

Definition 5.3. Given X and Y 1-reduced spaces with chosen points $x_0 \in X$ and $y_0 \in Y$, then the *wedge* $X \vee Y$ is the quotient of the disjoint union $X \amalg Y$ obtained by identifying x_0 and y_0 to a single point.

Consider the pushout diagram

$$\begin{tikzcd}
X \arrow[r, "i_1"] \arrow[d, "i_2"] & Y \arrow[d] \\
Z \arrow[r] & P
\end{tikzcd}$$

where X is the one-point simplicial set and Y, Z are 1-reduced simplicial sets and i_1, i_2 are morphism from X to the base point of Y and Z respectively. Then, $P_{(i_1,i_2)}$ is the wedge $Y \vee Z$.

Algorithm 8.
Input: two 1-reduced simplicial sets X and Y with effective homology.
Output: an equivalence $C(X \vee Y) \Lleftarrow DC(X \vee Y) \Rrightarrow HC(X \vee Y)$, where $C(X \vee Y)$ is the chain complex coming from the wedge of X and Y.

Definition 5.4. Given X and Y spaces, one can define the space of all lines segments joining points in X to points in Y. This is the *join* $X * Y$, the quotient space of $X \times Y \times I/\sim$, under the identifications $(x, y_1, 0) \sim (x, y_2, 0)$ and $(x_1, y, 1) \sim (x_2, y, 1)$. Thus we are collapsing the subspace $X \times Y \times \{0\}$ to X and $X \times Y \times \{1\}$ to Y.

Consider the pushout diagram

$$\begin{tikzcd}
X \times Y \arrow[r, "p_X"] \arrow[d, "p_Y"] & X \arrow[d] \\
Y \arrow[r] & P
\end{tikzcd}$$

where p_X and p_Y are the projections. Then, $P_{(p_X, p_Y)}$ is the join $X * Y$.

Algorithm 9.
Input: two simplicial sets X and Y with effective homology.
Output: an equivalence $C(X * Y) \Lleftarrow DC(X * Y) \Rrightarrow HC(X * Y)$, where $C(X * Y)$ is the chain complex coming from the join of X and Y.

6 Implementation

The algorithms explained in Section 5 have been implemented as a new module for the Kenzo system. The set of programs we have developed (with about 1600 lines) allows the computations of the pushout of two morphisms $f : X \to Y$ and $g : X \to Z$ when the effective homology of X, Y and Z is available.

In Subsubsection 6.1 we explain the essential part of these programs, describing the functions with the same format as in the Kenzo documentation [4]. In Subsubsection 6.2 we will see some examples of calculations.

6.1 A new module for the Kenzo system

In the development of the new module for Kenzo that allows one to compute the pushout associated with two morphism, the first step has been to define algorithms 2, 3, 4, 5 and 6 (the algorithm 1 is already implemented in the Kenzo system).
Direct Sum Algorithm (algorithm 2):

Let X and Y two simplicial sets, the programs described here builds $X \oplus Y$, an object with effective homology.

The generators of $X \oplus Y$ are represented internally in the system by a lisp object of the form:

```
(:direct-sum-gsm (d-sum-ind.d-sum-old))
```

where,

1. **d-sum-ind** is a non negative integer with value 0 or 1. 0 indicates the origin of the simplex is X and 1 for Y.
2. **d-sum-old** is a a non degenerate simplex, coming from one of the arguments X or Y as indicated by the value of d-sum-ind.

We have written several functions that allow us to construct the direct sum of two chain complexes. The description of some of these methods is shown here:

direct-sum-cmpr ***cmpr1 cmpr2*** *[Function]*

From the comparison functions *cmpr1* and *cmpr2*, build a comparison function to compare two generators of a direct sum.

direct-sum-basis ***basis1 basis2*** *[Function]*

From the functions *basis1* and *basis2* of the chain complexes X and Y, build a basis function for the direct sum of these chain complexes.

direct-sum-cmbn-split ***cmbn*** *[Function]*

From a combination of elements of the direct sum of X and Y, it returns two combinations the first one with elements of X and the second one with the elements of Y.

direct-sum-dffr ***dffr1 dffr2*** *[Function]*

From the lisp differential functions $dffr1$ and $dffr2$ of two chain complexes, build the lisp differential function of the direct sum of these chain complexes.

direct-sum ***chcm1 chcm2*** *[Method]*

Build the chain complex direct sum of the chain complexes *chcm1* and *chcm2*, using the basic functions above, as shown in the following call to `build-chcm`:

```
(the chain-complex
    (build-chcm
     :cmpr (direct-sum-cmpr (cmpr chcm1) (cmpr chcm2))
     :basis (direct-sum-basis (basis chcm1) (basis chcm2))
     :bsgn (direct-sum-gsm 0 (bsgn chcm1))
     :intr-dffr (direct-sum-dffr (dffr chcm1) (dffr chcm2))
     :strt :cmbn
     :orgn ‘(direct-sum ,chcm1 ,chcm2))
```

direct-sum-mrph ***sorc trgt mrph1 mrph2*** *[Function]*

Build the direct-sum of the morphisms *mrph1* and *mrph2*. This is a morphism of the same degree as *mrph1*. Return the morphism built by the following call to `build-mrph`:

```
(build-mrph
        :sorc sorc
        :trgt trgt
        :degr (degr mrph1)
        :intr (direct-sum-dffr mrph1 mrph2)
        :strt :cmbn
        :orgn `(direct-sum-mrph ,sorc ,trgt ,mrph1 ,mrph2))
```

direct-sum-efhm ***chcm1 chcm2*** *[Function]*

Build the homotopy equivalence of the direct sum of *chcm1* and *chcm2*.

Cone Algorithm (algorithm 3):

The cones are defined in the Kenzo system, but instead of using definition 3.10 it uses definition 6.1

Definition 6.1. Let C_* and D_* be two chain-complexes and $\phi : C_* \leftarrow D_*$ be a chain-complex morphism. Then the *cone* of ϕ denoted by $Cone(\phi)$ is the chain complex $Cone(\phi) = A_*$ defined as follows. First $A_n := C_n \oplus D_{n-1} \dots$

SES_2 Theorem (Theorem 3.14) works with Definition 3.10, then it has been necessary to implement this definition of the cone, of course based on the already implemented in Kenzo, as follows:

To distinguish to which chain complex belongs a generator in a combination of a cone, the following convention has been adopted: if `gc` is a generator of any degree of C, it will be represented in the cone by the list `(:con 0 gc)` and printed as `<CONE 0 gc>`. The symbol for D is `(:con 1 gd)`.

We have written several functions that allow us to construct the cone of a simplicial morphism. The description of some of these methods is shown here:

cone-cmpr ***cmpr0 cmpr1*** *[Function]*

From the 2 comparison functions *cmpr0* and *cmpr1*, build a comparison function adequate to compare the generators as represented in the cone

cone2-basis ***basis0 basis1*** *[Function]*

From the 2 basis function *basis0* and *basis1*, build a basis function for the cone. If at least one of the chain complex component of the cone is *locally effective*, the function returns the symbol `:locally-effective`

cone-3mrph-triangle-impl2 ***cmpr0 mrph0 mrph1 phi*** *[Function]*

Define the differential in the cone according to the formula:

$$d(cc, cd) = (d_C(cc), phi(cc) - d_D(cd))$$

cone2 ***mrph*** *[Method]*

Build Cone(*mrph*), using the above functions.

cone2-efhm ***mrph*** *[Function]*

From the formulas of the Cone Equivalence Theorem we can build the homotopy equivalence for the Cone(*mrph*).

AiBjC Algorithm (algorithm 4)

These algorithms were extracted from the GiftQ program of Francis Sergeraert, a Lisp program analogous to Kenzo but devoted to commutative algebra which is not yet www-available.

The main methods which allows the construction of the reduction $A \lll Cone(i)$ from the effective short exact sequence $0 \xleftarrow{0} A_* \underset{j}{\overset{\sigma}{\rightleftarrows}} B_* \underset{i}{\overset{\rho}{\rightleftarrows}} C_* \longleftarrow 0$ are :

AiBjC-RDCT-F ***A i rho B j sigma C*** *[Function]*

Build the f morphism of the reduction.

AiBjC-RDCT-G ***A i rho B j sigma C*** *[Function]*

Build the g morphism of the reduction.

AiBjC-RDCT-H ***A i rho B j sigma C*** *[Function]*

Build the h morphism of the reduction.

AiBjC-RDCT ***A i rho B j sigma C*** *[Function]*

Build the reduction using the basic functions above, as shown in the following call to `build-rdct`:

```
(build-rdct
     :f (AiBjC-rdct-f A i rho B j sigma C)
     :g (AiBjC-rdct-g A i rho B j sigma C)
     :h (AiBjC-rdct-h A i rho B j sigma C)
     :orgn '(AiBjC-rdct ,A ,i ,rho ,B ,j ,sigma ,C))
```

Remove Covers Algorithm (algorithm 5)

Let X a simplicial set, the programs described here builds the chain complex M coming from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled.

The generators of the chain complex coming from $X \times I$ are represented internally in the system by a lisp object of the form `(:crpr (dgop1.gmsm1).(dgop2.gmsm2))` where,

1. **dgop1** is an integer representing a coded degeneracy operator.
2. **gmsm1** is a non-degenerate simplex of X, to which is applied the degeneracy operator dgop1.
3. **dgop2** is an integer representing a coded degeneracy operator.
4. **gmsm2** is a non-degenerate simplex of I, to which is applied the degeneracy operator dgop2. I can be represented in Kenzo as the standard simplex of dimension 1, this Simplicial Set has two vertices represented as 1 and 2 and an edge 3.

The generators of the chain complex coming from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled are represented internally in the system by a lisp object of the form `(:crpr (dgop1.gmsm1).(dgop2.gmsm2))` where,

1. **dgop1** is an integer representing a coded degeneracy operator.
2. **gmsm1** is a non-degenerate simplex of X, to which is applied the degeneracy operator dgop1.
3. **dgop2** is an integer representing a coded degeneracy operator.
4. **gmsm2** is the non-degenerate simplex 3 of I, because the simplices of $X \times \{0\}$ and $X \times \{1\}$ are cancelled.

We have written several functions that allow us to construct the chain complex M coming from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled. The description of some of these methods is shown here:

remove-covers-basis ***basis1*** *[Function]*

From the function *basis1* of the simplicial set X, build a basis function for the chain complex from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled.

remove-covers-dffr ***dffr1*** *[Function]*

From the lisp differential functions $dffr1$ of a simplicial set, build the lisp differential function of the the chain complex from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled.

remove-covers ***smst1*** *[Function]*

Build the chain complex from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled, using the basic functions above and the comparison function of $X \times I$.

remove-covers-efhm ***smst1*** *[Function]*

Build the homotopy equivalence of the chain complex from $X \times I$ but with the simplices of $X \times \{0\}$ and $X \times \{1\}$ cancelled, using the functions from the AiBjC-rdct algorithm.

Suspension Algorithm (algorithm 6)

In Kenzo the suspension process is realized by the function `suspension`. This function has one argument, a reduced object, but if the object is not reduced, the result is undefined. Then, we have undertaken the task of developing the functor suspension described in Subsection 5.

suspension2-basis ***basis*** *[Function]*

From the functions *basis* of a chain complex, build a basis function for the suspension of this chain complex.

suspension2-dffr ***dffr*** *[Function]*

From the lisp differential functions $dffr$ of a chain complex, build the lisp differential function for the suspension of this chain complex.

suspension2 ***chcm*** *[Method]*

Build the chain complex suspension of the chain complex *chcm*, using the basic functions above.

suspension2-intr ***mrph*** *[Function]*

From a Kenzo morphism *mrph*, build an internal function corresponding to the suspension of the initial morphism.

suspension2 ***mrph*** *[Method]*

Build the suspension of the morphism *mrph*.

suspension2 ***rdct*** *[Method]*

Build the suspension of the reduction *rdct*.

suspension2 ***hmeq*** *[Method]*

Build the suspension of the homotopy equivalence *hmeq*.

Finally, the core of this new module consists in several functions that construct the pushout of two simplicial morphisms, implementing the algorithm 7.

The *non-degenerate* simplex of (f, g) are represented internally in the system by a lisp object of the form `(:pushout-gsm (p-ind.p-old))` where,

1. **p-ind** is a non negative integer with value 0, 1 or 2. p-ind indicates the origin of the simplex, 0 for $X \times I$, 1 for Y and 2 for Z.
2. **p-old** is a a non degenerate simplex of dimension, coming from one of the arguments $X \times I$ or Y or Z as indicated by the value of p-ind.

The main functions to build the pushout are:

pushout-cmpr ***cmpr1 cmpr2 cmpr3*** *[Function]*

From the comparison functions *cmpr1*, *cmpr2* and *cmpr2*, build a comparison function to compare two generators of a pushout.

pushout-basis ***basis1 basis2 basis3*** *[Function]*

From the functions *basis1*, *basis2* and *basis3* of the simplicial sets $X \times I$ Y and Z, build a basis function for the pushout of these simplicial sets.

pushout-face ***face1 face2 face3 f g*** *[Function]*

From the face operators *face1*, *face2* and *face3* and the morphism f and g, builds the face operator for the pushout.

pushout ***f g*** *[Method]*

Build the simplicial set pushout of the simplicial morphisms f and g, using the basic functions above, as shown in the following call to `build-smst`:

```
(let* ((X (sorc f))
       (XxI (crts-prdc X (delta 1)))
       (Y (trgt f))
       (Z (trgt g)))
  (the simplicial-set
    (let ((rslt (build-smst
                 :cmpr (pushout-cmpr (cmpr XxI) (cmpr Y) (cmpr Z))
                 :basis (pushout-basis (basis XxI) (basis Y) (basis Z))
                 :bspn (pushout-gsm 1 (bspn Y))
                 :face (pushout-face (face XxI) (face Y) (face Z) f g)
                 :orgn `(pushout ,f ,g))))
      (declare (type simplicial-set rslt))
      rslt)))
```

pushout-efhm ***f g*** *[Function]*

Applying the process explained in Subsection 5, this function returns the homotopy equivalence for the pushout.

For the implementation of algorithms 8 and 9 two functions have been developed:

wedge ***smst1 smst2*** *[Function]*

From the simplicial sets *smst1* and *smst2* build the wedge of them.

join ***smst1 smst2*** *[Function]*

From the simplicial sets *smst1* and *smst2* build the join of them.

To provide a better understanding of these new tools, some elementary examples of their use are shown in the next subsubsection.

6.2 Examples

In this subsection we present some examples of application of the programs we have developed for building the pushout of two morphisms. First, we consider the particular case of the wedge of Eilenberg Mac Lane spaces ($K(\mathbb{Z},2) \vee K(\mathbb{Z},2)$). As a second example, we will show the computation of the join of spheres. Finally, a sophisticated example giving a geometrical construction of $P^2(\mathbb{C})$ is presented.
Wedge of $K(\mathbb{Z},2)$ and $K(\mathbb{Z},2)$

```
> (cat-init) ✠
---done---
> (setf bkz (k-z 2)) ✠
[K13 Abelian-Simplicial-Group]
> (setf bkzwbkz (wedge bkz bkz)) ✠
[K41 Simplicial-Set]
> (homology bkzwbkz 0 9) ✠
Homology in dimension 0 :
Component Z
Homology in dimension 1 :

Homology in dimension 2 :
Component Z
Component Z
Homology in dimension 3 :

Homology in dimension 4 :
Component Z
Component Z
Homology in dimension 5 :

Homology in dimension 6 :
Component Z
Component Z
Homology in dimension 7 :

Homology in dimension 8 :
Component Z
Component Z
```

Join of S^3 and S^4

The join of the spheres S^n and S^m spheres is the sphere S^{n+m+1}.

```
> (cat-init) ✠
---done---
> (setf s3 (sphere 3)) ✠
[K1 Simplicial-Set]
> (setf s4 (sphere 4)) ✠
[K6 Simplicial-Set]
> (setf s3js4 (join s3 s4)) ✠
[K28 Simplicial-Set]
> (homology s3js4 0 9) ✠
Homology in dimension 0 :
Component Z
Homology in dimension 1 :

Homology in dimension 2 :

Homology in dimension 3 :

Homology in dimension 4 :

Homology in dimension 5 :

Homology in dimension 6 :

Homology in dimension 7 :

Homology in dimension 8 :
Component Z
```

That is the expected result from the definition of the sphere.

$P^2(\mathbb{C})$

This example gives a geometrical construction of $P^2(\mathbb{C})$. Take S^2 and construct the first stage of the Whitehead tower by doing:

```
> (cat-init) ✠
---done---
> (setf s2 (sphere 2)) ✠
[K1 Simplicial-Set]
> (setf ch2 (chml-clss s2 2)) ✠
[K12 Cohomology-Class on K1 of degree 2]
> (setf f2 (z-whitehead s2 ch2)) ✠
[K25 Fibration K1 -> K13]
> (setf x3 (fibration-total f2)) ✠
[K31 Simplicial-Set]
```

Then `x3` has the homotopy type of the 3-sphere S^3. More precisely `x3`$= s2 \times_{f2} K(Z,1)$ with `f2` an appropriate twisting function producing S^3 as a total space. It is easy to deduce a projection $f : X3 \rightarrow S2$. Taking the pushout of this f and the only map $g : X3 \rightarrow *$, then the pushout is $P^2(\mathbb{C})$. It can be checked that our programs obtain the right homology groups that are $(\mathbb{Z}, 0, \mathbb{Z}, 0, \mathbb{Z}, 0, 0, \ldots)$:

```
> (setf f (build-smmr :sorc x3 :trgt s2 :degr 0
                      :sintr #'(lambda (dmns gmsm)
                                 (declare (ignore dmns))
                              (absm (dgop1 gmsm) (gmsm1 gmsm)))
                      :orgn `(proj ,x3 ,s2))) ✠
[K36 Simplicial-Morphism K31 -> K1]
> (setf unipunctual (build-finite-ss '(x))) ✠
[K37 Simplicial-Set]
(setf g (build-smmr :sorc x3 :trgt unipunctual :degr 0
                    :sintr #'(lambda (dmns gmsm)
                               (if (and (equal dmns 0)
                                        (equal gmsm (bsgn x3)))
                                   'x
                                 nil))
                    :orgn `(proj ,x3 ,unipunctual))) ✠
[K42 Simplicial-Morphism K31 -> K37]
> (setf p (pushout f g)) ✠
[K53 Simplicial-Set]
> (homology p 0 10) ✠
Homology in dimension 0 :
Component Z
Homology in dimension 1 :

Homology in dimension 2 :
Component Z
Homology in dimension 3 :

Homology in dimension 4 :
Component Z
Homology in dimension 5 :

Homology in dimension 6 :

Homology in dimension 7 :

Homology in dimension 8 :

Homology in dimension 9 :
```

Bibliography

[1] R. Brown. The twisted Eilenberg-Zilber theorem. *Celebrazioni Archimedi de Secolo XX, Simposio di Topologia*, pages 34–37, 1967.

[2] G. Carlsson and R. J. Milgram. *Handbook of Algebraic Topology*, chapter Stable homotopy and iterated loop spaces, pages 505–583. North-Holland, 1995.

[3] J. P. Doeraene. Homotopy pull backs, homotopy push outs and joins. *Bulletin of the Belgian Mathematical Society*, 5:15–37, 1998.

[4] X. Dousson, J. Rubio, F. Sergeraert, and Y. Siret. The Kenzo program. Institut Fourier, Grenoble, 1998. `http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/`.

[5] S. Eilenberg and J. A. Zilber. Semi-simplicial complexes and singular homology. *Annals of Mathematics*, 51(3):499–513, 1950.

[6] P. Graham. *ANSI Common Lisp*. Prentice Hall, 1996.

[7] P. J. Hilton and S. Wylie. *Homology Theory*. Cambridge University Press, 1967.

[8] T. Kaczynski, K. Mischaikow, and M. Mrozek. *Computational Homology*, volume 157 of *Applied Mathematical Sciences*. Springer, 2004.

[9] S. MacLane. *Homology*. Springer, 1963.

[10] M. Mather. Pull-Backs in Homotopy Theory. *Canadian Journal of Mathematics*, 28(2):225–263, 1976.

[11] J. P. May. *Simplicial objects in Algebraic Topology*, volume 11 of *Van Nostrand Mathematical Studies*. 1967.

[12] J. Rubio and F. Sergeraert. Constructive Algebraic Topology. *Bulletin des Sciences Mathématiques*, 126(5):389–412, 2002.

[13] J. Rubio and F. Sergeraert. Constructive Homological Algebra and Applications, Lecture Notes Summer School on Mathematics, Algorithms, and Proofs. University of Genova, 2006. `http://www-fourier.ujf-grenoble.fr/~sergerar/Papers/Genova-Lecture-Notes.pdf`.

[14] W. Shih. Homologie des espaces fibrés. *Publications mathématiques de l'Institut des Hautes Études Scientifiques*, 13, 1962.