Effective computation of invariants of finite topological spaces

Julián Leonardo Cuevas Rozo

UNIVERSIDAD DE LA RIOJA

UNIVERSIDAD NACIONAL DE COLOMBIA

2021

Effective computation of invariants of finite topological spaces

Julián Leonardo Cuevas Rozo

Dissertation submitted for the degree of

Doctor of Philosophy

Advisors:
PhD. Laureano Lambán Pardo
PhD. Ana Romero Ibáñez
PhD. Humberto Sarria Zapata

UNIVERSIDAD
DE LA RIOJA

Universidad de La Rioja

Departamento de Matemáticas y Computación

Logroño, España

2021

Effective computation of invariants of finite topological spaces

Julián Leonardo Cuevas Rozo

Dissertation submitted for the degree of

Doctor of Science - Mathematics

Advisors:
PhD. Laureano Lambán Pardo
PhD. Ana Romero Ibáñez
PhD. Humberto Sarria Zapata

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Universidad Nacional de Colombia

Departamento de Matemáticas

Bogotá, Colombia

2021

A mis padres y mi hermano, quienes
han estado y estarán siempre conmigo.

A Liss, por haberme dado el regalo
más preciado de mi vida: Martín.

Resumen

Título en castellano: Cálculo efectivo de invariantes de espacios topológicos finitos

Hasta el momento, los métodos conocidos para el cálculo de invariantes de espacios topológicos finitos eran aplicables solamente a los posets de caras de complejos simpliciales o de CW-complejos regulares. En este trabajo hemos desarrollado versiones constructivas de algunos resultados teóricos de diferentes autores acerca de espacios finitos, produciendo en particular nuevos algoritmos para el cálculo explícito de algunos complejos de cadenas asociados a espacios finitos h-regulares y sus correspondientes generadores. Hasta donde sabemos, nuestro programa es el único software capaz de calcular grupos de homología de espacios topológicos finitos trabajando directamente sobre los posets. Hemos mejorado nuestros algoritmos sobre espacios finitos h-regulares mediante el uso de campos de vectores discretos, produciendo un nuevo algoritmo para construir dichos campos discretos definidos directamente sobre el poset, además de crear un proceso de h-regularización de espacios finitos permitiendo así ampliar la familia de espacios finitos h-regulares conocidos en la literatura.

También hemos presentado una interfaz entre los sistemas de álgebra computacional SageMath y Kenzo. Nuestro trabajo ha permitido que ambos sistemas colaboren mutuamente en algunos cálculos que no pueden ser hechos de manera independiente por dichos programas. Más aún, hemos creado un módulo en SageMath implementando espacios topológicos finitos y algunos conceptos relacionados. Finalmente, hemos considerado algunas estrategias para estudiar diferentes alternativas para calcular campos de vectores discretos de mayor longitud sobre espacios finitos, haciendo uso de algunas técnicas de aprendizaje automático para obtener campos de vectores discretos de la mayor longitud posible.

Palabras clave: Espacios topológicos finitos, grupos de homología, campos de vectores discretos, invariantes homotópicos, espacios finitos h-regulares, tipos de homotopía débil, topología computacional.

Abstract

Title in English: Effective computation of invariants of finite topological spaces

Up to now, the known methods for computing invariants of finite topological spaces were applicable only for face posets of simplicial complexes or regular CW-complexes. In this work, we have made constructive some theoretical results on finite topological spaces by different authors, producing in particular new algorithms for computing in an explicit way some chain complexes associated with h-regular finite topological spaces and their corresponding generators. Up to our knowledge, our new program is the only software able to compute homology groups of finite topological spaces working directly on the posets. We have improved our algorithms on h-regular spaces by using discrete vector fields, producing a new algorithm for constructing a discrete vector field defined directly on the poset; moreover, we have created a process of h-regularization of finite spaces, allowing to expand the family of h-regular finite spaces known in the literature.

We have presented an interface between the computer algebra systems SageMath and Kenzo. Our work has allowed both systems to collaborate in some computations which can not be done independently in any of the programs. Moreover, we have created a module implementing finite topological spaces and related concepts in SageMath. Finally, we have considered some strategies trying to study alternatives to compute longer discrete vector fields on finite spaces, considering some machine learning techniques to obtain discrete vector fields as big as possible.

Keywords: Finite topological spaces, homology groups, discrete vector fields, homotopic invariants, h-regular finite spaces, weak homotopy types, computational topology.

Acknowledgements

I want to express special thanks to my thesis advisors, who have been my guide in this process of development and completion of my doctoral studies. Many thanks to Ana Romero and Laureano Lambán for having supported me from the first day I arrived in Logroño. For your total willingness to provide me with the best possible environment to be able to concentrate on my studies. For being aware of my progress and for always pushing me to improve them. For your meticulous reviews of the presentations and writings that I have been making throughout these years, since without you I would have made many mistakes. I am thankful to Humberto Sarria, my Master's and PhD thesis advisor, who supported me in my studies from the beginning and to whom I owe the contact with the Universidad de La Rioja. Thank you so much for your recomendations and words of encouragement.

I am grateful to the Universidad Nacional de Colombia and especially to the professors of the Department of Mathematics for supporting me from my first years of professional training, providing me with the necessary tools to have undertaken this work. Thanks to Colciencias, for giving me the financial support I needed to be able to dedicate my efforts to the development of my doctoral studies.

Thanks to the Universidad de La Rioja for hosting me and to the entire Department of Mathematics and Computer Science since every one of its members has always been very kind and helpful. I am especially grateful to Julio Rubio for having come to Colombia at the right time when I was going to start my doctoral studies. Thanks to José Divasón for always being willing to help me solve my computational questions.

I also wish to thank to Adrián, Manuel and Jorge, my colleagues at the office, for many good times during working hours. I will always be thankful to you for your company. Thanks to Patricia, Ángela, Félix and all the PhD students of the Universidad de La Rioja for creating a very cool friendship environment.

I am and will always be very grateful to my parents and my brother, for being my source of inspiration and for being with me at all times, encouraging me to never give up. I hope I can be the son and brother that you are proud of. Thanks a lot to Liss, for her unconditional company, for being by my side since before traveling to Spain and not hesitating to support me, from the distance, on this trip. Finally, thanks to Martín, my beloved son, who turned out to be my faithful companion in the writing of this memoir.

Contents

Introduction

A *finite topological space* is a topological space whose underlying point set is finite. The study of finite topological spaces, or finite spaces in short, is a problem of great interest from a topological point of view, since they can be considered as models for a wide variety of spaces, including regular CW-complexes. Also, these spaces have been used to describe objects in image processing [Kov89], [LKE$^+$02] and for evaluating the topological inconsistency of geospatial data in [JBADB17]. Therefore, it is important to provide tools, both theoretical and computational, allowing to determine topological invariants of finite spaces.

In 1937, Alexandroff [Ale37] proved that finite topological spaces are related with preordered sets, connecting them by a one-to-one correspondence that, in particular, permits to establish an equivalence between finite spaces satisfying the T_0 axiom (T_0-spaces) and partially ordered sets (posets). This correspondence is very valuable, among other things, in the study of homotopy of finite spaces, because every finite space is homotopically equivalent to a T_0-space [Sto66], allowing to deal with T_0-spaces without altering homotopic properties. Some of the most important results on which the theory of finite topological spaces is based were established more than 50 years ago. The relation between finite spaces and posets permitted Stong [Sto66] to classify the homotopy types of finite spaces by means of an algorithmic way of deletion of some special points, called *beat points*, which reduces a finite space to the smallest homotopy equivalent space to it by means of operations called *strong collapses*. Moreover, the relation between finite spaces and posets makes appropriate the use of this structure for automatic computations, since the topological structure of a finite topological space can be represented in a suitable way from a computational point of view. At the same time, a fruitful relation between finite topological spaces and finite simplicial complexes is due to McCord [McC66], who assigns to each finite T_0-space X a simplicial complex $\mathcal{K}(X)$, the *order complex* of X, in such a way that X and $\mathcal{K}(X)$ are weak homotopy equivalent. Conversely, to each simplicial complex K, McCord relates a T_0-space $\mathcal{X}(K)$, the *face poset* of K, such that K and $\mathcal{X}(K)$ are weak homotopy equivalent. In addition, McCord proves that homotopy types of compact polyhedra are in one-to-one correspondence with weak homotopy types of finite topological spaces. At a first glance, the latter result is unexpected since non discrete finite spaces are non Hausdorff spaces (which in principle seems to convert them into uninteresting topological objects), unlike CW-complexes which do satisfy the Hausdorff property.

Fortunately, Alexandroff, Stong and McCord's results were not completely forgotten and served as the starting point of a deep research on Algebraic Topology of finite topological spaces. The first detailed exposition on the subject, based on the PhD Dissertation of Jonathan Barmak, can be found in [Bar11]. In that work, some well-known problems in Topology, Algebra and Geometry have been restated in terms of finite spaces. A Quillen's conjecture [Qui78] asserting that if the simplicial complex associated (in the sense of McCord) to the poset $S_p(G)$ of nontrivial p-subgroups of a finite group G, where p is a prime integer, is contractible then G has a nontrivial p-subgroup, was restated in terms of finite spaces as follows: $S_p(G)$ is contractible if and only if it is homotopically trivial. The Geometric Andrews-Curtis conjecture [AC65], which asserts that any contractible compact 3-polyhedron 3-deforms to a point, was translated to the setting of finite spaces as follows: a homotopically trivial finite T_0-space of height 2 3-deforms to a point. Indeed, Barmak's work allowed to enlarge the class of complexes which are known to satisfy this conjecture, by generalizing the concept of constructible complex to the notion of *quasi constructible 2-complex*.

Simple homotopy types of polyhedra were investigated by Whitehead [Whi50] by introducing the notions of *simplicial collapse* and *simplicial expansion*, moves that preserve the homotopy type of a complex. These concepts were translated to the theory of finite topological spaces by means of the notion of *weak point*, which was introduced by Barmak. The elementary move consisting on removing a weak point from a finite T_0-space corresponds exactly to a simplicial collapse of the associated simplicial complex (in the sense of McCord), allowing to find a one-to-one correspondence between simple homotopy types of finite spaces and simple homotopy types of finite simplicial complexes.

A general technique to analyze the topology of a simplicial complex was introduced by Forman with the name of Discrete Morse Theory [For98], a combinatorial adaptation of Morse Theory which is a powerful tool to study smooth manifolds. The main theorem of Discrete Morse Theory asserts that a simplicial complex is homotopy equivalent to a CW-complex with exactly one cell of dimension p for each critical simplex of dimension p with respect to a discrete Morse function defined on it. Moreover, Minian [Min12] introduced a version of discrete vector fields for posets that satisfy the *h-regularity property*; since the face poset of a simplicial complex is h-regular, Minian's results apply to any finite topological space coming from the simplicial context. Finally, recent works of Cianci and Ottina [CO17] have generalized Minian's results by defining an appropriate spectral sequence that converges to the homology of a finite T_0-space. Also, a wider class of *quasicellular spaces* (containing the h-regular spaces and the *cellular* spaces appearing in [Min12]) has been defined and it has been shown that the homology of such spaces can be computed by means of a concrete chain complex ([CO17]).

Algebraic Topology provides a linearization of topological problems, which usually present a pure non-linear nature. Then, the pursuit of reasonable linear algebra algorithmic solutions is of great interest from a computational point of view. In particular, our interest consists in providing symbolic computation systems in topology with tools devoted to finite topological spaces, such that it is possible to compute topological invariants of these spaces, which includes in an implicit way computations over a wide variety of topological spaces. To implement our methods we consider the symbolic computation system Kenzo [DRSS99], which was developed by Francis Sergeraert and some coworkers and allows the user to work with chain complexes and to compute homotopy and homology groups of simplicial sets.

This work is organized as will be described in the next lines. Chapter 1 contains some notations, definitions and previous results about Algebraic Topology on finite topological spaces, their connections with simplicial complexes and some insights into the Discrete Morse Theory which will be useful along our work. Also, a presentation of the Kenzo program as a powerful tool for the development of our algorithmic implementations in posterior chapters, is included.

In Chapter 2 we present a new module for the Kenzo program allowing to compute some invariants of finite topological spaces. In this way, we have enhanced Kenzo with new topological objects and powerful functionalities. Identification of beat points and weak points permits to apply methods of point reductions to find weak homotopy equivalent spaces to a given finite space in Kenzo. Moreover, we have developed algorithms which are based on new constructive versions of results found in [Min12] and [CO17] in order to determine homology groups of h-regular finite spaces without constructing the simplicial complex associated with the poset. The implementation of an algorithm to compute homologically admissible Morse matchings in order to use the Discrete Morse Theory to study the topology of finite spaces is also presented.

Once we have developed algorithms to compute invariants on h-regular finite spaces, a question arises naturally: how can we construct h-regular spaces in order to apply our results? In fact, given a finite T_0-space, its barycentric subdivision (the poset of the simplices ordered by inclusion of the simplicial complex associated to the finite T_0-space) is an h-regular space. In Chapter 3, we present a new technique to modify a finite space of height at most 2 in order to obtain an h-regular space which is simple homotopy equivalent to the initial one, with a smaller number of elements and edges than the barycentric subdivision.

We believe it is necessary to bring symbolic computation systems closer to their potentially users. This is particularly important in the case of Kenzo, due to its nearness to Lisp. In this line, in Chapter 4 we describe a new interface we have created to integrate the Kenzo program and the computer algebra system SageMath [Dev20], in order to transform topological data structures from one system to the other one in such a way that both systems can collaborate together in computations. Regarding to finite topological spaces, we will show a new module that we have created for finite spaces in SageMath by means of the developed interface.

In order to achieve better reductions in the size of the objects involved in computations, mainly chain complexes, in Chapter 5 we have considered some different strategies to compute discrete vector fields on finite spaces, by means of changing the way the candidates to vectors are chosen. In this way, we try to obtain a discrete vector field as big as possible. Moreover, we have explored some machine learning techniques such as reinforcement learning and Monte-Carlo tree search, obtaining promising results.

The memoir ends with a chapter which includes conclusions and further work, and the bibliography.

Contributions and participation in conferences

Software productions

- We have enhanced Kenzo with a new module of finite topological spaces. Available at `https://github.com/jcuevas-rozo/finite-topological-spaces/`.

- We have developed an interface between the computer algebra systems SageMath and Kenzo. Available at `https://trac.sagemath.org/ticket/27880`. After our participation in the *Google Summer of Code 2020*, this interface has been improved with new functionalities available at `https://trac.sagemath.org/ticket/29879`.

- A module of finite topological spaces for SageMath has been developed and submitted. Available at `https://trac.sagemath.org/ticket/30400`.

Publications and preprints

- Cuevas-Rozo J., Lambán L., Romero A. and Sarria H., *Effective homological computations on finite topological spaces*, Applicable Algebra in Engineering, Communication and Computing (2020), DOI: 10.1007/s00200-020-00462-8.

- Cuevas-Rozo, J., Divasón, J., Marco-Buzunáriz, M. and Romero, A., *A Kenzo interface for algebraic topology computations in SageMath*. ACM Commun. Comput. Algebra **53** (2), 61–64 (2019), DOI: 10.1145/3371991.3371999.

- Cuevas-Rozo J., Lambán L., Romero A. and Sarria H., *h-regularization of finite topological spaces*, Preprint, available here.

- Cuevas-Rozo J., Divasón J., Marco-Buzunáriz M. and Romero A., *Integration of the Kenzo system within SageMath for Algebraic Topology Computations*, Preprint, available here.

Presentations in conferences

- *Computing spectral sequences with Kenzo and SageMath* (joint work with Jose Divasón, Miguel Marco-Buzunáriz and Ana Romero) at Women in Algebra and Symbolic Computations, December 2019, Bad Dürkheim, Germany.

- *h-regularización de espacios topológicos finitos* at VIII Encuentro de Jóvenes Topólogos - VIII EJT, October 2019, Santiago de Compostela, Spain.

- *A Kenzo Interface for Algebraic Topology Computations in SageMath* (joint work with Jose Divasón, Miguel Marco-Buzunáriz and Ana Romero) at 44th International Symposium on Symbolic and Algebraic Computation - ISAAC 2019, July 2019, Beijing, China.

  * It was distinguished with the *Best Software Demonstration Award* sponsored by Fachgruppe Computeralgebra and Maplesoft http://issac-conference.org/2019/.

- *Computing with Kenzo from Sage* (joint work with Miguel Marco-Buzunáriz and Ana Romero) at Effective Methods in Algebraic Geometry - MEGA 2019, June 2019, Madrid, Spain.

- *Cálculo de invariantes topológicos sobre espacios topológicos finitos* (joint work with Laureano Lambán, Ana Romero and Humberto Sarria) at XXII Congreso Colombiano de Matemáticas - CCM 2019, June 2019, Popayán, Colombia.

- *New algorithms for computing homology of finite topological spaces* at 24th Conference on Applications of Computer Algebra - ACA 2018, June 2018, Santiago de Compostela, Spain.

- *Point reduction algorithms and discrete vector fields for finite topological spaces in the Kenzo system* at Fourth International School on Computer Algebra and its Applications - EACA 2018, March 2018, Santiago de Compostela, Spain.

Chapter 1

Preliminaries

In this chapter we set some notations and definitions about finite topological spaces, simplicial complexes, chain complexes and some topics about Discrete Morse Theory as well as basic results relating them, that we will use along this work. For further reading about finite topological spaces see [Bar11], for Discrete Morse Theory see [For98] and for general concepts in Algebraic Topology see [Hat02], [Mun84], [Rot98]. Moreover, since the algorithms we are going to design in our work will be included in the Kenzo computer algebra system [DRSS99], some basic ideas about this program will be presented.

1.1 Finite topological spaces

1.1.1 Finite spaces and posets

The main objects of study in this work are the finite topological spaces. We assume the reader is familiar with basic notions in topology such as topological space, subspace, open and closed sets, quotient space and closures. Throughout this work, the n-dimensional sphere will be denoted by S^n and the symbol $*$ will be used to denote a one-point set.

Definition 1.1. A ***finite space*** X is a topological space whose underlying point set is finite.

Observe that in these spaces, arbitrary intersections of open sets are always open, so that the next definition makes sense.

Definition 1.2. The intersection of all the open sets that contain an element $x \in X$, denoted by U_x^X, is called the ***minimal open set*** of x in X (if the finite space in which we are working is clear from the context, the symbol U_x is used to denote the minimal open set of x). The tilded version is defined by $\widehat{U}_x = U_x \setminus \{x\}$.

The collection $\mathcal{U}_X = \{U_x\}_{x\in X}$ is a basis for the topology of a finite space X, that is, it satisfies that $X = \cup_{x\in X} U_x$ and the intersection of any two elements $V, W \in \mathcal{U}_X$ is a union of elements in $\mathcal{U}_X$. It is easy to see that the collection $\mathcal{U}_X$ is the minimal basis of the finite space X in the following sense: if $\mathcal{B}$ is a basis for the topology of X then $\mathcal{U}_X \subseteq \mathcal{B}$.

In the context of finite spaces, the subspaces U_x play a crucial role in the description of topological properties. Other subspaces that are important in the study of finite spaces are the closures of singletons.

Definition 1.3. Given a finite space X and an element $x \in X$, the ***closure*** of $\{x\}$ is denoted by F_x^X (the tilded version is defined by $\widehat{F}_x = F_x \smallsetminus \{x\}$). The ***star*** of x is $C_x = U_x \cup F_x$ and its tilded version $\widehat{C}_x = C_x \smallsetminus \{x\}$ is called the ***link*** of x.

It is easy to see that the closed sets of a finite topological space define a topology on it.

Definition 1.4. Let X be a finite space. The ***opposite*** of X, denoted by X^{op}, is the finite space whose underlying set is X and whose topology is given by the basis $\mathcal{F}_X = \{F_x^X\}_{x \in X}$. Observe that $U_x^{X^{op}} = F_x^X$ for all $x \in X$.

In topology, there exist some properties that pretend to distinguish disjoint sets and distinct points, known as *separation axioms*. Among them, the most important one, in the context of finite topological spaces, is the T_0 axiom. We recall the corresponding definition.

Definition 1.5. A topological space X is a T_0***-space*** (also called a ***Kolmogorov space***) if for any two different points x and y there is an open set which contains one of these points and not the other one (x and y are *topologically distinguishable*).

Example 1.6. On the set $X = \{a, b, c\}$, consider the topologies $\mathcal{T}_1 = \{\emptyset, \{a\}, X\}$ and $\mathcal{T}_2 = \{\emptyset, \{a\}, \{a, b\}, \{a, c\}, X\}$. The finite space $(X, \mathcal{T}_2)$ is a T_0-space while $(X, \mathcal{T}_1)$ is not (observe that b and c are not topologically distinguishable).

The close relationship between finite topological spaces and partially ordered sets (posets) plays a crucial role in the theory of finite topological spaces.

Definition 1.7. Let X be a finite poset. A subset $C \subseteq X$ is a ***chain*** if it is a totally ordered subset of X, that is, $x \leqslant y$ or $y \leqslant x$ for every pair of elements $x, y \in C$. The ***height*** of an element $x \in X$ is defined as $h(x) = \max\{\#C : C \text{ is a chain and } x = \max(C)\} - 1$ and the ***height*** of X is the number $h(X) = \max\{h(x) : x \in X\}$. A ***fence*** is a finite sequence of points $x_0, x_1, \ldots, x_n \in X$ such that x_i and x_{i+1} are comparable, for all $0 \leqslant i \leqslant n-1$.

Definition 1.8. Let X be a poset. The ***Hasse diagram*** of X, denoted by $\mathcal{H}(X)$ is a digraph whose vertices are the points of X and whose edges are the ordered pairs (x, y) such that $x < y$ and there exists no z such that $x < z < y$ (x is the ***tail*** and y is the ***head*** of the edge (x, y)). The set of edges is denoted by $\mathrm{E}(\mathcal{H}(X))$.

Example 1.9. Let $X = \{a, b, c, d, e\}$ be a set and consider the relation on X given by $\Delta \cup \{(a, b), (a, c), (a, d), (b, d), (e, b), (e, d)\}$ where $\Delta = \{(x, x) : x \in X\}$. Then X, together with the given relation, is a poset whose Hasse diagram is represented in Figure 1.1. In this case, $h(a) = h(e) = 0$, $h(b) = h(c) = 1$ and $h(X) = h(d) = 2$.

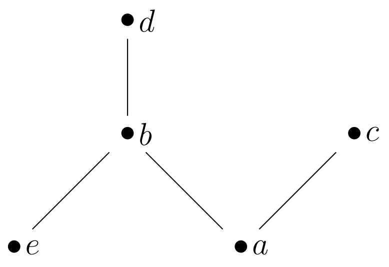


Figure 1.1: Hasse diagram of the poset in Example 1.9.

There exists a well-known result due to Alexandroff [Ale37] providing a one-to-one correspondence between finite spaces and finite preordered sets. If $(X, \leqslant)$ is a finite preordered set, the family of subsets $\{y \in X : y \leqslant x\}_{x \in X}$ is a basis for a topology on X. Conversely, if X is a finite space, a preorder relation can be defined as follows:

$$x \leqslant y \Longleftrightarrow x \in U_y \Longleftrightarrow U_x \subseteq U_y. \tag{1.1}$$

Moreover, in [Ale37] it is shown that the separation axiom T_0 corresponds to the anti-symmetry property, so that finite T_0-spaces are equivalent to finite posets. If X is a finite T_0-space, we also denote by X the associated (equivalent) poset.

The preorder relation of X^{op} is the inverse preorder of X, therefore by (1.1) we can write:

$$U_x^X = \{y \in X : y \leqslant x\},\ F_x^X = \{y \in X : x \leqslant y\} \tag{1.2}$$

When we write a finite space X in roster notation $X = \{x_1, \ldots, x_n\}$, we use the symbol U_k (resp. F_k) to denote U_{x_k} (resp. F_{x_k}).

The relation between finite topological spaces and posets is functorial, that is, it extends to the corresponding morphisms: continuous functions and order preserving maps.

Some of the principal topological properties that are used to distinguish topological spaces are connectedness and the stronger condition path-connectedness.

Definition 1.10. A topological space X is ***connected*** if the only subsets of X which are both open and closed are X and the empty set. The ***connected components*** of X are the maximal connected subsets (ordered by inclusion). X is ***path-connected*** if for every pair of points $x, y \in X$, there exists a ***path*** $\alpha : I = [0, 1] \longrightarrow X$ (a continuous map from I to X) such that $\alpha(0) = x$ and $\alpha(1) = y$.

It is well known that, in arbitrary topological spaces, the notions of connectedness and path-connectedness are not equivalent properties (indeed, if a space is path-connected then it is connected, but the converse is not true in general). However, in the context of finite spaces these concepts are in fact equivalent and coincide with the order-connected notion on preordered sets (a preordered set X is ***order-connected*** if for every pair $x, y \in X$, there exists a fence starting in x and ending in y).

Proposition 1.11. [Bar11] *Let X be a finite space. Then the following are equivalent:*

1. *X is a connected topological space.*

2. *X is an order-connected preorder.*

3. *X is a path-connected topological space.*

1.1.2 Basic constructions

We introduce now some topological constructions which are useful in order to describe the structure of finite spaces.

Definition 1.12. Let X and Y be two finite T_0-spaces. The ***non-Hausdorff join*** (also called ***ordinal sum***) of X and Y, denoted by $X \circledast Y$, is the disjoint union $X \sqcup Y$ with the topology associated to the following order: one keeps the respective orderings within X and Y and one sets $x \leqslant y$ for every $x \in X$ and $y \in Y$. The ***non-Hausdorff cone*** of X is defined by $\mathbb{C}(X) = X \circledast *$. The ***non-Hausdorff suspension*** of X is the finite space $\mathbb{S}(X) = X \circledast S^0$; defining $\mathbb{S}^0(X) = X$, the ***non-Hausdorff suspension of order*** n of X is defined recursively by $\mathbb{S}^n(X) = \mathbb{S}(\mathbb{S}^{n-1}(X))$.

Note that for every x in a finite T_0-space, its link can be written as $\widehat{C}_x = \widehat{U}_x \circledast \widehat{F}_x$.

In general, the quotient of a T_0-space by a subspace is not a T_0-space. The next result shows a necessary and sufficient condition for which the quotient of a finite T_0-space by a subspace $A \subsetneq X$ is again a T_0-space. The *open hull* of A is $\underline{A} = \cup_{a \in A} U_a^X$ and its closure is $\overline{A} = \cup_{a \in A} F_a^X$.

Lemma 1.13. [Bar11] *Let A be a subspace of a finite T_0-space X, then X/A is T_0 if and only if $\underline{A} \cap \overline{A} = A$.*

The minimal open subsets of a quotient space are characterized by the next result.

Lemma 1.14. [Bar11] *Let $x \in X$ and let $q : X \longrightarrow X/A$ be the quotient map. If $x \in \overline{A}$, $U_{q(x)} = q(U_x \cup \underline{A})$. If $x \notin \overline{A}$, $U_{q(x)} = q(U_x)$.*

A particular case of a quotient space is the wedge construction.

Definition 1.15. Let X and Y be finite T_0-spaces and let $x_0 \in X$, $y_0 \in Y$. The ***wedge*** of X and Y is the finite T_0-space $X \vee Y := X \sqcup Y/\{x_0, y_0\}$.

Observe that if X and Y are finite T_0-spaces and $x_0 \in X$, $y_0 \in Y$, the set $A = \{x_0, y_0\}$ seen as a subspace of $X \sqcup Y$ satisfies $A = \underline{A} \cap \overline{A}$. Then, the wedge is a T_0-space.

Example 1.16. In Figure 1.2 are shown two finite T_0-spaces X and Y which we use to illustrate the non-Haussdorf cone of X, the non-Hausdorff suspension of Y, the non-Hausdorff joins $X \circledast Y$ and $Y \circledast X$ and the wedge $X \vee Y = X \sqcup Y/\{c, w\}$.

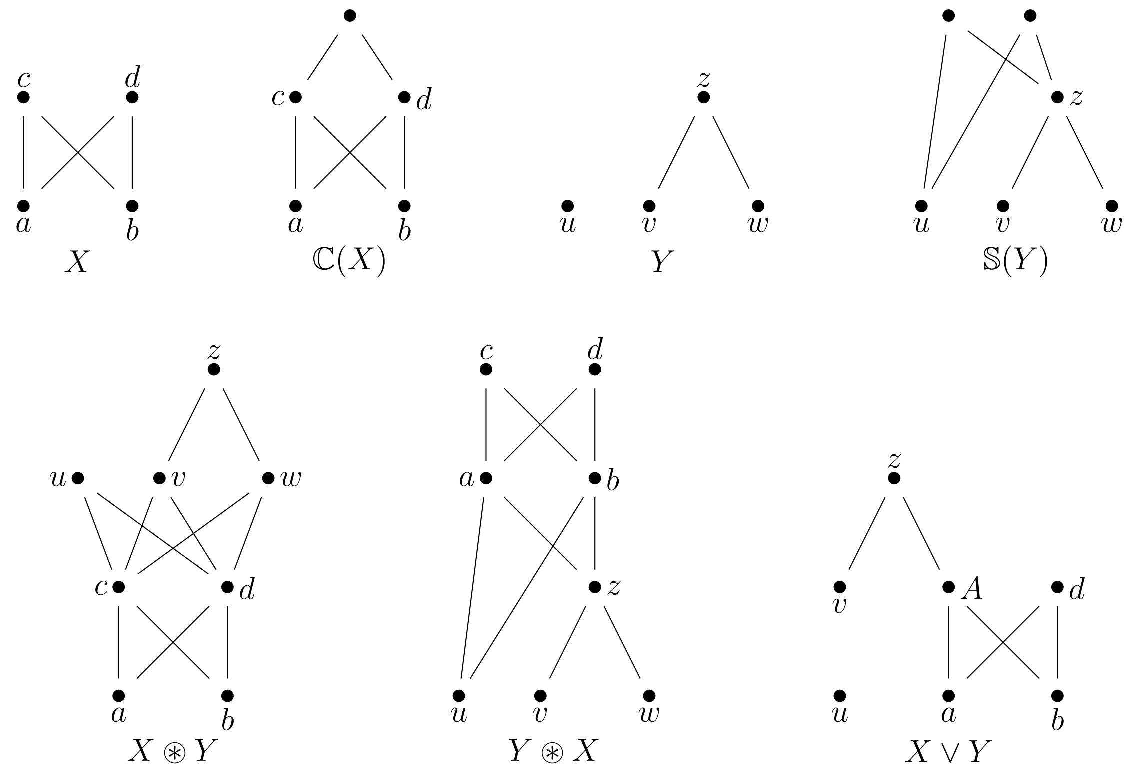


Figure 1.2: Some basic constructions involving finite T_0-spaces.

1.1.3 Topogenous and Stong matrices

The matricial representation of a finite topological space is given by topogenous and Stong matrices.

Definition 1.17. Given a finite space $X = \{x_1, \dots, x_n\}$, its ***topogenous matrix*** is the $n \times n$ matrix $T_X = [t_{ij}]$ defined by

$$t_{ij} = \begin{cases} 1 & , \ x_i \in U_j \\ 0 & , \ x_i \notin U_j \end{cases}. \tag{1.3}$$

Observe that the minimal open set of x_k (resp. closure of $\{x_k\}$) can be *read* from the column (resp. row) k of the topogenous matrix by considering the row (resp. column) indexes equal to 1. In symbols,

$$U_k = \{i : t_{ik} = 1\}, \ F_k = \{j : t_{kj} = 1\} \tag{1.4}$$

Definition 1.18. Given a finite T_0-space $X = \{x_1, \dots, x_n\}$, its ***Stong matrix*** is the $n \times n$ matrix $S_X = [s_{ij}]$ satisfying

$$s_{ij} = \begin{cases} 1 & , \ x_i \in U_j \text{ and there is no } k \text{ such that } U_i \subsetneq U_k \subsetneq U_j \\ 0 & , \ \text{in other case.} \end{cases} \tag{1.5}$$

In [Shi68], Shiraki introduced the term *topogenous matrix* to denote the transpose of the matrix that we have called topogenous above. We prefer this choice since the topogenous matrix (as defined in Definition 1.17), in the case of T_0-spaces, can be regarded as the incidence matrix corresponding to the partial order induced by the finite T_0-space and the Stong matrix is the adjacency matrix of the Hasse diagram of the associated poset.

Remark 1.19. Observe that the enumeration chosen for the elements in a finite space X plays a crucial role when defining a topogenous matrix. Each finite space with n elements has $n!$ associated topogenous matrices, each of them corresponding to a unique permutation of the elements of the space. For this reason, each time we mention *the* topogenous matrix of X, an enumeration of the elements of X must be fixed.

Topogenous matrices can be characterized by means of the next result.

Theorem 1.20. [Kri66] *Let $X = \{x_1, \dots, x_n\}$ be a finite T_0-space and let $T_X = [t_{ij}]$ be its topogenous matrix. Then T_X satisfies the following properties for all $1 \leqslant i, j, k \leqslant n$:*

1. $t_{ij} \in \{0, 1\}$.
2. $t_{ii} = 1$.
3. *If $t_{ij} = 1$ and $t_{jk} = 1$ then $t_{ik} = 1$.*

Conversely, if an $n \times n$ matrix $T = [t_{ij}]$ satisfies the above properties, then T induces a topology on X whose minimal open sets are $U_i = \{x_k : t_{ki} = 1\}$ for all i.

Bearing in mind the permutation-similarity relation between two square matrices (A and B are *permutation-similar* if there exists a permutation matrix P such that $A = P^T BP$), the next result characterizes the homeomorphisms between finite spaces by means of their topogenous matrices.

Theorem 1.21. [Shi68] *Let X and Y be finite spaces with topogenous matrices T_X and T_Y, respectively. Then X and Y are homeomorphic if and only if T_X and T_Y are permutation-similar.*

As we have seen, the T_0 separation axiom in finite spaces corresponds to the antisymmetry property of the preorder given in (1.1). In terms of topogenous matrices, these properties are equivalent to the upper triangular property (up to homeomorphism).

Theorem 1.22. [Shi68] *A finite space X is T_0 if and only if its topogenous matrix T_X is permutation-similar to an upper triangular topogenous matrix.*

Remark 1.23. By Theorem 1.22, from now on, when we say X is a finite T_0-space, we will assume that the enumeration of its elements was chosen in such a way that its topogenous matrix T_X is upper triangular.

1.1.4 Homotopy types and weak homotopy types

We recall some well-known notions and results in algebraic topology and explain them in the finite topological space framework.

Definition 1.24. Let X and Y be topological spaces, $A \subseteq X$ and let $f, g : X \longrightarrow Y$ be continuous functions. The maps f and g are ***homotopic relative to*** A if there exists a continuous map (a ***homotopy***) $H : I \times X \longrightarrow Y$ (I is the unit interval) such that:

1. $H(0, x) = f(x)$ and $H(1, x) = g(x)$ for all $x \in X$,
2. $H(t, x) = f(x) = g(x)$ for all $(t, x) \in I \times A$.

In this case we write $f \simeq g \ (rel\ A)$; when $A = \emptyset$, f and g are said to be ***homotopic*** and it is denoted by $f \simeq g$.

It can be proved that if X and Y are finite spaces, there is a natural correspondence between the set of homotopies $\{H : I \times X \longrightarrow Y\}$ and the set of paths $\{\alpha : I \longrightarrow Y^X\}$, where Y^X is the set of continuous maps from X to Y equipped with the compact-open topology which corresponds (by Alexandroff correspondence) to the preorder relation on Y^X given by $f \leqslant g$ if $f(x) \leqslant g(x)$ for every $x \in X$ (see details in [Bar11]). As a consequence of these considerations, the next result is obtained.

Proposition 1.25. [Bar11] *Let X and Y finite spaces, $A \subseteq X$ and let $f, g : X \longrightarrow Y$ be two maps. Then $f \simeq g \ (rel\ A)$ if and only if there exists a finite sequence of maps $\{f_i\}_{i=0}^n$ such that $f = f_0 \leqslant f_1 \geqslant f_2 \leqslant \cdots \geqslant f_n = g$ and $f_i|_A = f|_A$ for every $0 \leqslant i \leqslant n$.*

Intuitively, two continuous functions are homotopic if one can be *continuously deformed* into the other (the continuous deformation is given by the homotopy between them).

Definition 1.26. Let X and Y be topological spaces. X and Y are ***homotopy equivalent*** (X and Y have the same ***homotopy type***), if there exist maps (***homotopy equivalences***) $f : X \longrightarrow Y$ and $g : Y \longrightarrow X$ such that $g \circ f \simeq id_X$ and $f \circ g \simeq id_Y$. In this case we write $X \overset{\text{he}}{\approx} Y$. A space X is ***contractible*** if it has the homotopy type of a one-point space (this is equivalent to say that $id_X \simeq c$ for some constant function $c : X \longrightarrow *$) i.e. $X \overset{\text{he}}{\approx} *$.

A finite T_0-space X with maximum satisfies that the identity map id_X is comparable with the constant map $c(x) = \max(X)$ for every $x \in X$ since $id_X \leqslant c$, then $id_X \simeq c$, by Proposition 1.25 and therefore X is contractible. A similar argument shows that a finite T_0-space with minimum is contractible. In particular, the subspaces U_x and F_x are contractibles.

Definition 1.27. Let X and Y be topological spaces, $A \subseteq X$ and let $i_A : A \hookrightarrow X$ be the inclusion map. The subspace A is said to be a:

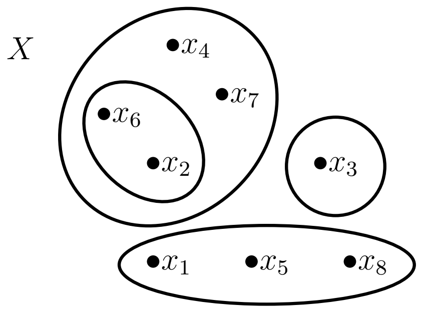
1. ***retract*** of X if there exists a continuous map (a ***retraction***) $r : X \longrightarrow A$ such that $r \circ i_A \simeq id_A$.

2. ***deformation retract*** of X if there exists a retraction r such that $i_A \circ r \simeq id_X$.
3. ***strong deformation retract*** of X if there exists a retraction r such that $i_A \circ r \simeq id_X$ $(rel\ A)$.

Observe that $X \overset{\text{he}}{\approx} A$ when A is a deformation retract or a strong deformation retract of X. In the setting of finite topological spaces, Stong proved that every finite space is homotopy equivalent to a finite T_0-space, therefore when studying homotopy types of finite spaces, we can restrict ourselves to finite T_0-spaces. This fact is established in the following result.

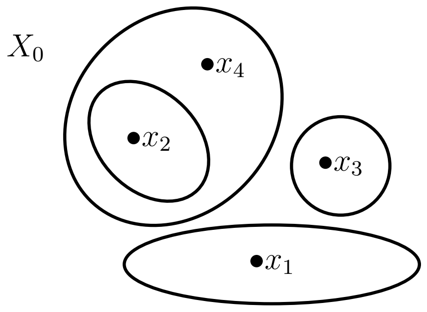
Proposition 1.28. [Sto66] *Let X be a finite space. Let X_0 be the quotient space $X/\sim$ where $x \sim y$ if $x \leqslant y$ and $y \leqslant x$. Then X_0 is T_0 and the quotient map $q : X \longrightarrow X_0$ is a homotopy equivalence. Moreover, X_0 is a strong deformation retract of X.*

Example 1.29. The next diagram represents the minimal open sets of a finite space X. Its topogenous matrix T_X is also shown.



$$T_X = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Observe that X is not T_0 since $U_2 = U_6 = \{x_2, x_6\}$. Then, by taking one representative of each equivalence class $[x_1] = \{x_1, x_5, x_8\}$, $[x_2] = \{x_2, x_6\}$, $[x_3] = \{x_3\}$ and $[x_4] = \{x_4, x_7\}$, we obtain the finite T_0-space X_0 as in Proposition 1.28. Note that in T_X, the corresponding columns and rows of the elements in each equivalence class are the same; for example, columns (and rows) 2 and 6 are equal in T_X. By removing repeated columns (or rows) in T_X, we can obtain the topogenous matrix of the finite T_0-space X_0 [CR16] (observe that it is upper triangular).



$$T_{X_0} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

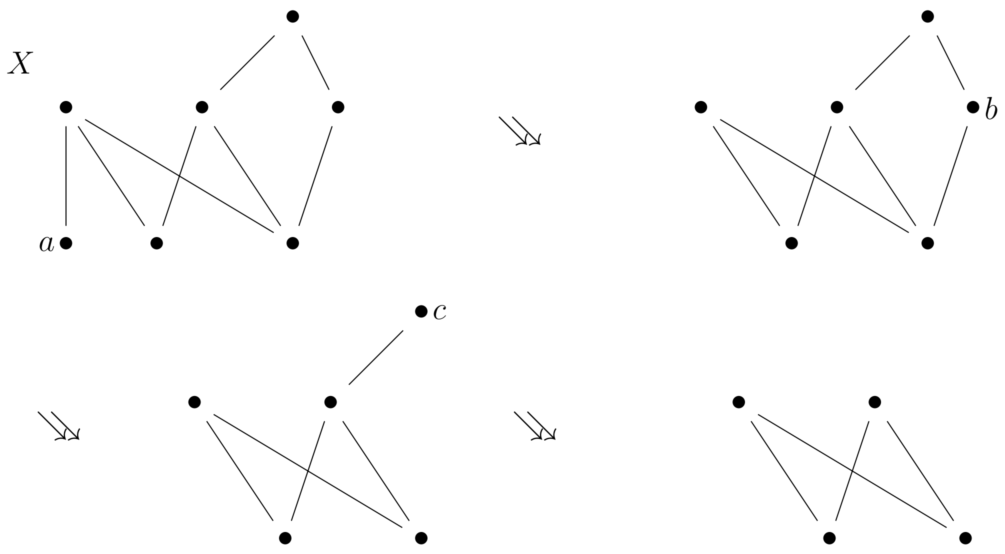
In the setting of finite T_0-spaces, there is a fundamental move that allows to reduce the size of a space in order to obtain a homotopy equivalent space with minimum cardinality. Such a move is the elimination of beat points.

Definition 1.30. Let X be a finite T_0-space. A point $x \in X$ is a ***down beat point*** if $\widehat{U}_x$ has a maximum; x is an ***up beat point*** if $\widehat{F}_x$ has a minimum. In any of these cases, x is a ***beat point*** of X and it is said that there is an ***elementary strong collapse*** from X to $X \smallsetminus \{x\}$. There is a ***strong collapse*** $X \searrow\!\!\searrow Y$ (or a ***strong expansion*** $Y \nearrow\!\!\nearrow X$) if there is a sequence of elementary strong collapses starting in X and ending in Y. X is a ***minimal finite space*** if it has no beat points.

Definition 1.31. A ***core*** of X is a strong deformation retract of X which is a minimal finite space.

Stong, in [Sto66], called *linear* and *colinear points* to the beat points; the term *beat point* was introduced in some unpublished notes written by Peter May for REU 2003, which are now summarized in [May20]. Beat points can be identified in the Hasse diagram of a finite T_0-space X as follows: a point $x \in X$ is a down beat point if and only if there is only one edge with x at its head; x is an up beat point if and only if there is only one edge with x at the tail.

Example 1.32. The following sequence of elementary strong collapses shows a procedure to obtain a core of a finite T_0-space X by removing beat points. Observe that a is an up beat point of X, b is an up beat point and a down beat point of $X \smallsetminus \{a\}$ and, finally, c is a down beat point of $X \smallsetminus \{a, b\}$. Then, $X \smallsetminus \{a, b, c\}$ is a minimal finite space.



The removing of beat points allows to find strong deformation retracts of a finite T_0-space. Indeed, Barmak proved that this is the only procedure to find strong deformation retracts from a finite T_0-space.

Proposition 1.33. [Sto66] *Let X be a finite T_0-space and let $x \in X$ be a beat point. Then, the inclusion map $i : X \smallsetminus \{x\} \longrightarrow X$ is a homotopy equivalence.*

Proposition 1.34. [Bar11] *Let X be a finite T_0-space and let $A \subseteq X$. Then, $X \searrow\!\!\searrow A$ if and only if A is a strong deformation retract of X.*

Homotopy types of finite spaces are characterized by classes of homeomorphism of minimal finite spaces.

Theorem 1.35. [Sto66] *A homotopy equivalence between minimal finite spaces is a homeomorphism. In particular the core of a finite space is unique up to homeomorphism and two finite spaces are homotopy equivalent if and only if they have homeomorphic cores.*

As a consequence of the above result, observe that in the context of finite spaces, a space is contractible if and only if its cores are one-point spaces.

Regarding finite topological spaces, weak homotopy types are fundamental to understand the relations between finite spaces and simplicial complexes (as we will see in Section 1.2). See [Hat02] for the definition of *homotopy groups*, which are one important invariant for topological spaces.

Definition 1.36. Let X and Y be finite spaces. A continuous map $f : X \longrightarrow Y$ is a ***weak homotopy equivalence*** if it induces an isomorphism between the homotopy groups of both spaces i.e. for every base point $x \in X$, $f_* : \pi_i(X, x) \longrightarrow \pi_i(Y, f(x))$ is a group isomorphism for all $i \geqslant 1$ and $f_* : \pi_0(X) \longrightarrow \pi_0(Y)$ is a bijective function. X and Y are ***weak homotopy equivalent*** (X and Y have the same ***weak homotopy type***) if there exists a finite sequence of topological spaces $X = X_1, X_2, \ldots, X_n = Y$ and weak homotopy equivalences $X_i \to X_{i+1}$ or $X_{i+1} \to X_i$ for each $1 \leqslant i \leqslant n-1$. In this case we write $X \overset{\text{we}}{\approx} Y$. A space X is ***homotopically trivial*** if it has the weak homotopy type of a one-point space i.e. $X \overset{\text{we}}{\approx} *$.

Definition 1.37. Given a topological space Z, a ***finite model*** of Z is a finite space X such that $X \overset{\text{we}}{\approx} Z$. X is said to be a ***minimal finite model*** (of Z) if it is a finite model of minimum cardinality.

Example 1.38. McCord [McC66] proved that the finite T_0-space $\mathbb{S}^n(S^0)$ is a finite model of the n-dimensional sphere S^n for every $n \geqslant 0$. In fact, $\mathbb{S}^n(S^0)$ is a minimal finite model of S^n [Bar11]. Minimal finite models of some spheres are shown in Figure 1.3.

A useful way to check if a continuous map $f : X \longrightarrow Y$ between finite T_0-spaces is a weak homotopy equivalence is verifying that such a map is a local weak homotopy equivalence over a *basis like open cover* of the codomain, that is, a basis for a topology in the underlying set of Y, perhaps different from the original topology. In particular, the minimal basis $\mathcal{U}_Y = \{U_y\}_{y \in Y}$ is a basis like open cover of Y and this is what is used most times.

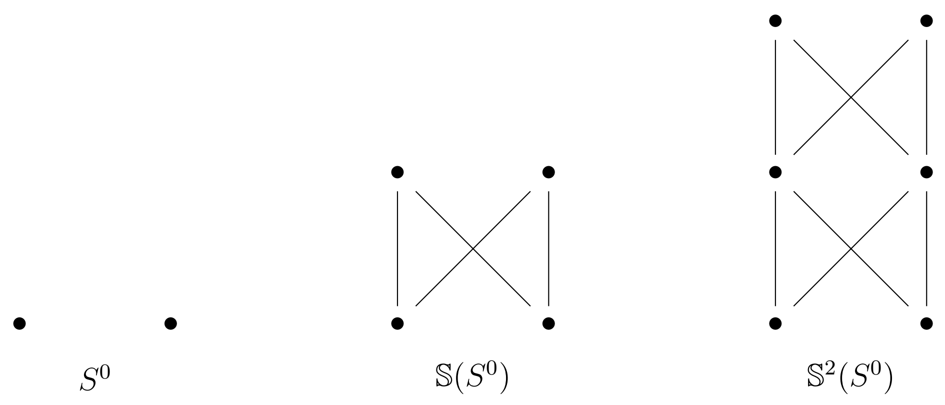


Figure 1.3: Minimal finite models for S^0, S^1 and S^2.

Theorem 1.39. [McC66] *Let X and Y be topological spaces and let $f : X \longrightarrow Y$ be a continuous map. Suppose that there exists a basis like open cover $\mathcal{U}$ of Y such that each restriction*

$$f|_{f^{-1}(U)} : f^{-1}(U) \longrightarrow U$$

is a weak homotopy equivalence for every $U \in \mathcal{U}$. Then $f : X \longrightarrow Y$ is a weak homotopy equivalence.

1.2 Simplicial complexes and simple homotopy types

The relationship between finite topological spaces and simplicial complexes is crucial in order to compute invariants on finite topological spaces.

1.2.1 Simplicial complexes

We recall some basic notions on simplicial complexes.

Definition 1.40. Let V a finite set. A ***simplicial complex*** K is a family of subsets of V satisfying the following conditions:

1. $\{v\} \in K$ for all $v \in V$.
2. If $\tau \in K$ and $\sigma \subseteq \tau$ then $\sigma \in K$.

The set V is called the ***vertex set*** of K and the elements of K are called ***simplices***. Given a simplex $\sigma \in K$, the ***dimension*** of σ is $\dim(\sigma) = \#\sigma - 1$. If the dimension of σ is p, we say that σ is a p-simplex; the ***p-skeleton*** of K is the collection of p-simplices of K and it is denoted by K_p. In particular, the vertex set V is in a one-to-one correspondence with K_0, so that by abuse of notation, we write $v \in V$ when $\{v\} \in K$. The ***dimension*** of K is the supremum of the dimensions of its simplices. If $\tau \subseteq \sigma$, we say that τ is a ***face*** of σ

and if $\tau \subset \sigma$, τ is called a ***proper face*** of σ. A ***facet*** in K is any simplex that is not a face of any larger simplex.

Given a p-simplex $\sigma = \{v_0, v_1, \dots, v_p\}$, the ***closed simplex*** $\overline{\sigma}$ is the set of formal convex combinations of the vertices of σ i.e. $\overline{\sigma} = \{\sum_{i=0}^{p} \alpha_i v_i : \alpha_i \geqslant 0, \sum_{i=0}^{p} \alpha_i = 1\}$. A metric d (and therefore a topology induced by d) can be defined on a closed simplex by the formula

$$d\left(\sum_{i=0}^{p} \alpha_i v_i, \sum_{i=0}^{p} \beta_i v_i\right) = \left(\sum_{i=0}^{p} (\alpha_i - \beta_i)^2\right)^{1/2} \tag{1.6}$$

The ***geometric realization*** $|K|$ of a simplicial complex K is the set

$$|K| = \bigcup_{\sigma \in K} \overline{\sigma} = \left\{ \sum_{v \in K} \alpha_v v : \alpha_v \geqslant 0, \sum_{v \in K} \alpha_v = 1, \{v : \alpha_v > 0\} \in K \right\}$$

A ***polyhedron*** is the geometric realization of a simplicial complex and a ***triangulation*** of a polyhedron X is a simplicial complex K whose geometric realization is homeomorphic to X.

The ***support*** of a point $x = \sum_{v \in V} \alpha_v v \in |K|$ is the set $\text{support}(x) = \{v : \alpha_v > 0\}$. $|K|$ is a topological space with the topology given by the collection $\{U \subseteq |K| : U \cap \overline{\sigma} \text{ is open in } \overline{\sigma}\}$. In the particular case K is a ***finite simplicial complex*** (a simplicial complex with a finite number of vertices), the topology of $|K|$ coincides with the topology induced by the metric d defined in (1.6) and $|K|$ can be imbedded in $\mathbb{R}^n$ for some $n \in \mathbb{N}$.

Given a simplicial complex K, its ***barycentric subdivision*** K' is the simplicial complex whose vertices are the simplices of K, and the simplices of K' are the sets of simplices $\{\sigma_0, \sigma_1, \dots, \sigma_n\}$ satisfying $\sigma_0 \subset \sigma_1 \subset \cdots \subset \sigma_n$. The ***barycenter*** of a simplex $\sigma \in K$ is the point $b(\sigma) = \sum_{v \in \sigma} \frac{v}{\#v} \in |K|$.

Let K and L be simplicial complexes. A ***simplicial map*** $\varphi : K \longrightarrow L$ is a vertex map $V_K \longrightarrow V_L$ such that $\sigma \in K$ implies $\varphi(\sigma) \in L$. A simplicial map $\varphi : K \longrightarrow L$ induces a continuous map $|\varphi| : |K| \longrightarrow |L|$ defined by $|\varphi|(\sum_{v \in K} \alpha_v v) = \sum_{v \in K} \alpha_v \varphi(v)$.

Regarding finite topological spaces, McCord [McC66] associates to each finite T_0-space a simplicial complex and for each finite simplicial complex it was associated a finite T_0-space allowing to prove that weak homotopy types of finite spaces coincide with homotopy types of finite CW-complexes.

Definition 1.41. Let X be a finite T_0-space. The ***simplicial complex associated to*** X is the simplicial complex whose simplices are the nonempty chains of X and it is denoted by $\mathcal{K}(X)$ (it is also called the ***order complex*** of X).

If $f : X \longrightarrow Y$ is a continuous map between finite T_0-spaces, the associated map $\mathcal{K}(f) : \mathcal{K}(X) \longrightarrow \mathcal{K}(Y)$, defined by $\mathcal{K}(f)(x) = f(x)$, is a simplicial map.

Definition 1.42. Let K be a finite simplicial complex. The ***finite*** T_0***-space associated to*** K is the poset of simplices of K ordered by inclusion and it is denoted by $\mathcal{X}(K)$ (it is also called the ***face poset*** of K).

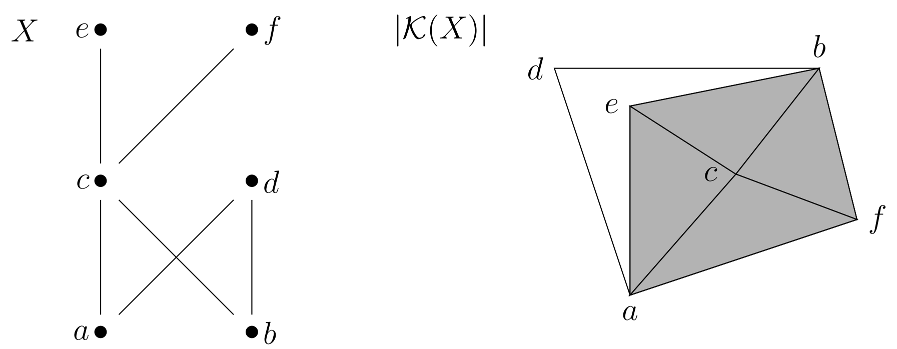


Figure 1.4: A finite T_0-space X and the geometric realization of its order complex $\mathcal{K}(X)$.

If $\varphi : K \longrightarrow L$ is a simplicial map between finite simplicial complexes, the associated map $\mathcal{X}(\varphi) : \mathcal{X}(K) \longrightarrow \mathcal{X}(L)$, defined by $\mathcal{X}(\varphi)(\sigma) = \varphi(\sigma)$, is a continuous map.

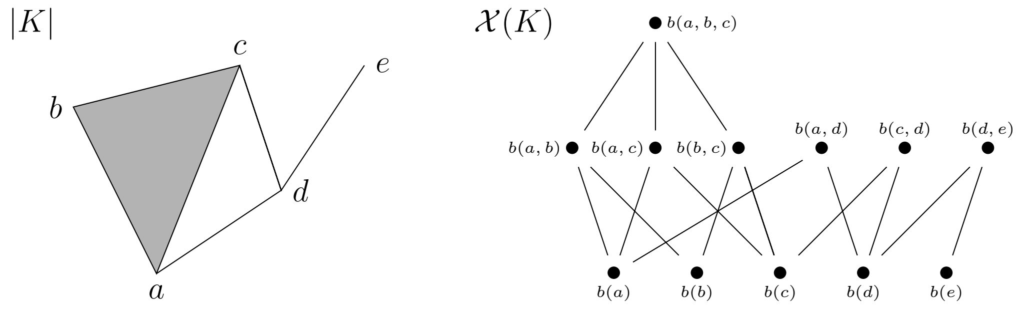


Figure 1.5: The geometric realization $|K|$ of a finite simplicial complex K and its face poset $\mathcal{X}(K)$.

In Figures 1.4 and 1.5 are illustrated the order complex of a finite T_0-space and the face poset of a finite simplicial complex. Note that if K is a finite simplicial complex, $\mathcal{K}(\mathcal{X}(K))$ is the (first) barycentric subdivision of K i.e. $\mathcal{K}(\mathcal{X}(K)) = K'$. By analogy, one has the following definition.

Definition 1.43. The ***barycentric subdivision*** of a finite T_0-space X is defined as the finite T_0-space $X' = \mathcal{X}(\mathcal{K}(X))$.

An important result establishes that finite simplicial complexes and finite T_0-spaces have the same weak homotopy types by means of the McCord maps.

Definition 1.44. Let X be a finite T_0-space and let K be a finite simplicial complex. The ***$\mathcal{K}$-McCord map*** $\mu_X : |\mathcal{K}(X)| \longrightarrow X$ is defined by $\mu_X(x) = \min(\text{support}(x))$. The ***$\mathcal{X}$-McCord map*** is the composite $\mu_K = \mu_{\mathcal{X}(K)} s_K^{-1} : |K| \longrightarrow \mathcal{X}(K)$, where $s_K : |K'| \longrightarrow |K|$ is the homeomorphism defined by $s_K(\sigma) = b(\sigma)$.

Theorem 1.45. [McC66] *The $\mathcal{K}$-McCord map μ_X is a weak homotopy equivalence for every finite T_0-space X. The $\mathcal{X}$-McCord map μ_K is a weak homotopy equivalence for every finite simplicial complex K.*

For the sake of completeness, we mention some definitions about CW-complexes which are related to finite topological spaces and simplicial complexes. See [Hat02], [Mun84] for more information.

Definition 1.46. Let K be a CW-complex. K is a ***regular*** complex if for each cell e^n, the attaching map $S^{n-1} \longrightarrow K$ is a homeomorphism onto its image $\dot{e}^n$, the boundary of e^n. K is an ***h-regular*** complex if the attaching map of each cell is a homotopy equivalence with its image and the closed cells $\overline{e^n}$ are subcomplexes of K (by [Bar11, Corollary 7.1.4], a CW-complex is h-regular if the closed cells are contractible subcomplexes).

Observe that regular complexes are h-regular. On an h-regular complex K, an order relation on the cells of K can be defined by $e \leqslant e'$ if $e \subseteq \overline{e'}$. The ***face poset*** $\mathcal{X}(K)$ is the poset of cells of K ordered by the face relation $\leqslant$.

Theorem 1.47. [Bar11] *If K is a finite h-regular complex, $\mathcal{X}(K)$ is a finite model of K.*

For the class of finite topological spaces, the concept of *h-regular space* was introduced in [Min12]. This particular type of finite spaces will be of great interest for our work.

Definition 1.48. A finite T_0-space X is called ***h-regular*** if for every $x \in X$, the order complex $\mathcal{K}(\widehat{U}_x)$ is homotopy equivalent to the $(h(x)-1)$-dimensional sphere i.e. $\widehat{U}_x \overset{\text{we}}{\approx} S^{h(x)-1}$.

The face poset $\mathcal{X}(K)$ of any h-regular CW-complex K (in particular, of any finite simplicial complex) is h-regular. In [Min12], Figure 1.6 is shown as an example of an h-regular space which is not the face poset of a regular CW-complex.

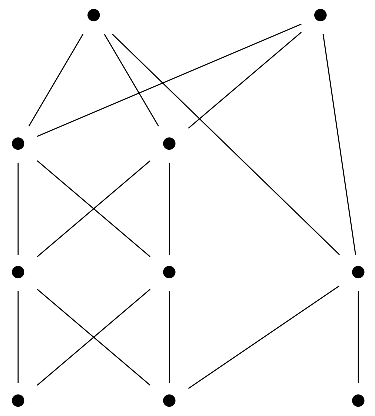


Figure 1.6: An h-regular poset (shown in [Min12, Figure 1]).

1.2.2 Simple homotopy types

Collapses are a useful tool for reducing simplicial complexes preserving their topological invariants.

Definition 1.49. Let K be a finite simplicial complex. There is an ***elementary simplicial collapse*** from K to a subcomplex L if there are only two simplices $\sigma, \tau \in K \smallsetminus L$ such that τ is the unique simplex of K containing σ properly (σ is called a ***free face*** of τ). In symbols, $K = L \cup a\sigma$ and $L \cap a\sigma = a\dot{\sigma}$ where $\sigma \in K$ and a is a vertex of K not in σ ($\dot{\sigma}$ is the boundary of σ and $a\dot{\sigma}$ is a simplicial cone).

We say that K (simplicially) ***collapses*** to L (or that L ***expands*** to K) if there exists a sequence of elementary simplicial collapses starting in K and ending in L. This is denoted by $K \searrow L$ or $L \nearrow K$. Two finite simplicial complexes K and L are ***simple homotopy equivalent*** if there is a sequence $K = K_1, K_2, \dots, K_n = L$ such that for each $1 \leqslant i < n$, $K_i \searrow K_{i+1}$ or $K_i \nearrow K_{i+1}$. We denote this by $K \diagup\!\!\searrow L$.

Theorem 1.50. [Bar11] *Let X be a finite T_0-space and let $x \in X$. If the inclusion map $i : X \smallsetminus \{x\} \longrightarrow X$ is a weak homotopy equivalence, it induces a simple homotopy equivalence $|\mathcal{K}(X \smallsetminus \{x\})| \longrightarrow |\mathcal{K}(X)|$. In particular $X \smallsetminus \{x\} \diagup\!\!\searrow X$.*

Barmak found an elementary move in the setting of finite spaces which corresponds exactly to a simplicial collapse of the associated polyhedra; such a move consists of removing a weak point of the space.

Definition 1.51. Let X be a finite T_0-space. A point $x \in X$ is a ***down weak point*** if $\widehat{U}_x$ is contractible; x is an ***up weak point*** if $\widehat{F}_x$ is contractible. In any of these cases, x is a ***weak point*** of X and in this case we will say that there is an ***elementary collapse*** from X to $X \smallsetminus \{x\}$. There is a ***collapse*** $X \searrow Y$ (or an ***expansion*** $Y \nearrow X$) if there is a sequence of elementary collapses starting in X and ending in Y. Two finite T_0-spaces X and Y are ***simple homotopy equivalent*** if there is a sequence $X = X_1, X_2, \dots, X_n = Y$ of finite T_0-spaces such that for each $1 \leqslant i < n$, $X_i \searrow X_{i+1}$ or $X_i \nearrow X_{i+1}$. We denote this by $X \diagup\!\!\searrow Y$. A space X is ***collapsible*** if it collapses to a one-point space i.e. $X \diagup\!\!\searrow *$.

Observe that beat points are weak points and it is easy to see that $x \in X$ is a weak point if and only if its link $\widehat{C}_x$ is contractible. Elimination of weak points of a finite T_0-space does not modify the weak homotopy type.

Proposition 1.52. [Bar11] *Let X be a finite T_0-space and let $x \in X$ be a weak point. Then, the inclusion map $i : X \smallsetminus \{x\} \longrightarrow X$ is a weak homotopy equivalence.*

A series of fundamental results about homotopy types, weak homotopy types and simple homotopy types of finite T_0-spaces and finite simplicial complexes can be summarized in the diagram in Figure 1.7.

$$\begin{tikzcd}
X \overset{he}{\simeq} Y \arrow[r, Rightarrow] & X \mathrel{\diagup\!\!\!\searrow} Y \arrow[r, Rightarrow] \arrow[d, Leftrightarrow] & X \overset{we}{\approx} Y \arrow[d, Leftrightarrow] & \\
 & \mathcal{K}(X) \mathrel{\diagup\!\!\!\searrow} \mathcal{K}(Y) \arrow[r, Rightarrow] & {|\mathcal{K}(X)| \overset{we}{\approx} |\mathcal{K}(Y)|} \arrow[r, Leftrightarrow] & {|\mathcal{K}(X)| \overset{he}{\simeq} |\mathcal{K}(Y)|}
\end{tikzcd}$$

$$\begin{tikzcd}
\mathcal{X}(K) \overset{he}{\simeq} \mathcal{X}(L) \arrow[r, Rightarrow] & \mathcal{X}(K) \mathrel{\diagup\!\!\!\searrow} \mathcal{X}(L) \arrow[r, Rightarrow] \arrow[d, Leftrightarrow] & \mathcal{X}(K) \overset{we}{\approx} \mathcal{X}(L) \arrow[d, Leftrightarrow] & \\
 & K \mathrel{\diagup\!\!\!\searrow} L \arrow[r, Rightarrow] & {|K| \overset{we}{\approx} |L|} \arrow[r, Leftrightarrow] & {|K| \overset{he}{\simeq} |L|}
\end{tikzcd}$$

Figure 1.7: Relation between simple homotopy types, weak homotopy types and homotopy types of finite T_0-spaces and finite simplicial complexes (taken from [Bar11]).

1.3 Effective homology and Kenzo

The primitive structures in the Kenzo system are simplicial sets and chain complexes. As we have presented, there is a relationship between finite topological spaces and simplicial complexes. Then, a direct method to compute in Kenzo some topological invariants of a finite space X is through the chain complex which can be canonically defined from its associated simplicial complex $\mathcal{K}(X)$. Usually, the size of this chain complex is too large and it is imperative to use some kind of reduction techniques. In this section we introduce some notions and results related to these issues. Moreover we include a brief introduction to the Kenzo system.

1.3.1 Chain complexes and homology

Definition 1.53. A ***chain complex*** $C_* = (C_p, d_p)$ is a sequence of abelian groups and homomorphisms

$$\cdots \longrightarrow C_{p+1} \xrightarrow{d_{p+1}} C_p \xrightarrow{d_p} C_{p-1} \longrightarrow \cdots$$

such that $d_p d_{p+1} = 0$ for each $p \in \mathbb{Z}$. The homomorphism d_p is called the ***differential map*** or ***boundary map*** of degree p. The condition $d_p d_{p+1} = 0$ is equivalent to the inclusion $\text{im } d_{p+1} \subseteq \ker d_p$. The elements of $\ker d_p$ are called ***cycles*** and the elements of $\text{im } d_{p+1}$ are called ***boundaries***. The p-th ***homology group*** of this chain complex is the quotient group

$$H_p(C_*) = \ker d_p / \text{im } d_{p+1}$$

An important example of chain complexes are those who generate the so-called *simplicial homology groups*.

Definition 1.54. An ***oriented simplicial complex*** K is a simplicial complex and a partial order on K_0 whose restriction to the vertices of any simplex in K is a linear order.

Note that every linear ordering of K_0 makes K into an oriented simplicial complex.

Definition 1.55. Let K be an oriented simplicial complex of dimension n. Define $C_p(K)$ as the free abelian group with basis all symbols $[q_0, \dots, q_p]$, where $\{q_0, \dots, q_p\}$ is a p-simplex in K and $q_0 < q_1 < \cdots < q_p$. Define the boundary map $d_p : C_p(K) \longrightarrow C_{p-1}(K)$ by extending by linearity the formula

$$d_p([q_0, \dots, q_p]) = \sum_{i=0}^{p} (-1)^i [q_0, \dots, \widehat{q_i}, \dots, q_p] \tag{1.7}$$

where $\widehat{q_i}$ means delete the vertex q_i. It can be proved that

$$0 \longrightarrow C_n(K) \xrightarrow{d_n} \cdots \xrightarrow{d_2} C_1(K) \xrightarrow{d_1} C_0(K) \longrightarrow 0$$

is a chain complex, denoted by $C_*(K) = (C_p(K), d_p)$. The p-th homology group of $C_*(K)$, $H_p(K) := H_p(C_*(K))$, is called the p-th ***simplicial homology group*** of K. The p-th ***reduced simplicial homology group*** $\tilde{H}_p(K)$ is the p-th homology group of the augmented chain complex

$$0 \longrightarrow C_n(K) \xrightarrow{d_n} \cdots \xrightarrow{d_2} C_1(K) \xrightarrow{d_1} C_0(K) \xrightarrow{\varepsilon} \mathbb{Z} \longrightarrow 0$$

where ε is the homomorphism $\varepsilon(\sum_i n_i \sigma_i) = \sum_i n_i$ for $\sigma_i \in C_0(K)$.

It is easy to see that for all $p \geqslant 1$, $H_p(K) \cong \tilde{H}_p(K)$ and $H_0(K) \cong \tilde{H}_0(K) \oplus \mathbb{Z}$. Simplicial homology groups can be regarded as a particular case of a more general construction on arbitrary topological spaces, called ***singular homology groups*** (see [Hat02], [Rot98]).

The simplicial homology groups of a simplicial complex K are isomorphic to the singular homology groups of its geometric realization, that is, $H_p(K) \cong H_p(|K|)$, for all $p \in \mathbb{Z}$ (see [Rot98]).

Regarding finite topological spaces, note that $H_p(X) \cong H_p(|\mathcal{K}(X)|) \cong H_p(\mathcal{K}(X))$ and $H_p(K) \cong H_p(|K|) \cong H_p(\mathcal{X}(K))$ for every finite T_0-space X and for every finite simplicial complex K, taking into account Theorem 1.45 and the fact that a weak homotopy equivalence induces isomorphisms on homology groups (see [Hat02]). Then, the homology groups of a finite T_0-space can be computed by using techniques applicable to simplicial complexes. However, since the simplices of $\mathcal{K}(X)$ are the chains of X, in [CO17] a chain complex expressed entirely in terms of X and its chains is defined, which can be used to describe the homology groups of X in a more efficient way.

Definition 1.56. Let X be a finite T_0-space. For $p \in \mathbb{N}$, a ***p-chain*** of X is a chain of X of cardinality $p+1$. The empty chain is regarded as a (-1)-chain. The notation $[v_0, \dots, v_p]$ is used for a p-chain $\{v_0, \dots, v_p\}$ of X with $v_{i-1} < v_i$ for all $1 \leqslant i \leqslant p$. The set of p-chains of X is denoted by $\mathrm{Ch}_p(X)$ and the set of chains of X is $\mathrm{Ch}(X)$.

Definition 1.57. Let X be a finite T_0-space. The ***$\mathcal{P}$-chain complex*** associated to X is the chain complex $C_*^{\mathcal{P}}(X) = (C_p^{\mathcal{P}}(X), d_p^{\mathcal{P}})$ where $C_p^{\mathcal{P}}(X)$ is the free abelian group with basis the set $\mathrm{Ch}_p(X)$ of p-chains of X for $p \geqslant 0$ and $C_p^{\mathcal{P}}(X) = 0$ for $p < 0$ and the boundary map $d_p^{\mathcal{P}} : C_p^{\mathcal{P}}(X) \longrightarrow C_{p-1}^{\mathcal{P}}(X)$ is defined by extending by linearity the formula

$$d_p^{\mathcal{P}}([v_0, \dots, v_p]) = \sum_{i=0}^{p} (-1)^i [v_0, \dots, \widehat{v}_i, \dots, v_p]$$

where $[v_0, \dots, \widehat{v}_i, \dots, v_p]$ is the $(p-1)$-chain of X obtained by removing the point v_i from $[v_0, \dots, v_p] \in \mathrm{Ch}_p(X)$. The p-th homology group $H_p^{\mathcal{P}}(X) := H_p(C_*^{\mathcal{P}}(X))$ is called the p-th ***P-homology group*** of the finite T_0-space X.

The above definitions permit to obtain the next result.

Proposition 1.58. *[CO17] Let X be a finite T_0-space. Then $H_p^{\mathcal{P}}(X) \cong H_p(X)$ for all $p \in \mathbb{Z}$.*

1.3.2 Effective homology and discrete vector fields

A very useful technique to analyze the topology of simplicial complexes is Discrete Morse Theory [For98]. In particular, we are interested in a concrete application of this theory which provides "reductions" of chain complexes preserving homology and which can be applied in a very direct way to finite topological spaces.

Definition 1.59. An ***algebraic cellular complex*** (ACC) is a family $C_* = (C_p, d_p, \beta_p)$ where (C_p, d_p) is a chain complex and every C_p is provided with a distinguished $\mathbb{Z}$-basis β_p; every basis component $\sigma \in \beta_p$ is called a p-cell.

Let us introduce now the main notions of *the effective homology method*, introduced in [Ser94] and deeply explained in [RS06]. These notions are implemented in Kenzo.

Definition 1.60. A ***reduction*** ρ is a diagram

$$\rho = \boxed{\begin{tikzcd} \widehat{C}_* \arrow[loop left, "h"] \arrow[r, shift right, "f"'] & C_* \arrow[l, shift right, "g"'] \end{tikzcd}} \tag{1.8}$$

where:

1. The nodes $\widehat{C}_*$ and C_* are chain complexes.

2. The arrows f and g are chain complex morphisms.

3. The self-arrow h is a homotopy operator of degree 1.

4. The following relations are satisfied:

$$\begin{aligned} fg &= \mathrm{id}_{C_*} \\ gf + dh + hd &= \mathrm{id}_{\widehat{C}_*} \\ fh &= 0 \\ hg &= 0 \\ hh &= 0 \end{aligned} \tag{1.9}$$

Following [RS10], we will denote a reduction by $\rho = (f, g, h) : \widehat{C}_* \Rrightarrow C_*$. Reductions play a relevant role in the task of computing homology. If a reduction $\rho : \widehat{C}_* \Rrightarrow C_*$ is defined, properties (1.9) ensure that the homology groups of the complexes $\widehat{C}_*$ and C_* are canonically isomorphic.

Definition 1.61. A ***(strong chain) equivalence*** $\varepsilon \equiv (C_* \Lleftarrow\!\!\Rrightarrow E_*)$ between two chain complexes C_* and E_* is a triple (D_*, ρ, ρ') where D_* is a chain complex and ρ and ρ' are reductions from D_* over C_* and E_* respectively: $C_* \overset{\rho}{\Lleftarrow} D_* \overset{\rho'}{\Rrightarrow} E_*$.

Definition 1.62. An ***effective chain complex*** C_* is an algebraic cellular complex where each group C_n is finitely generated.

Definition 1.63. An ***object with effective homology*** is a triple (X, EC_*, ε) where X is an object (e.g. a simplicial set, a topological space) possessing a canonically associated free chain complex $C_*(X)$, EC_* is an effective chain complex and ε is an equivalence between $C_*(X)$ and EC_*, that is, $C_*(X) \overset{\varepsilon}{\Lleftarrow\!\!\Rrightarrow} EC_*$.

The notion of object with effective homology makes it possible to compute homology groups of complicated spaces by using effective complexes and computing their homology groups, which can be easily obtained through some elementary algorithms on matrices. Let us observe that the chain complex $C_*(X)$ could be so big that its homology groups are not directly computable (for example, when the complex is not of finite type).

The following result provides a particular type of reduction for a given chain complex.

Theorem 1.64. [RS10] *Let $C_* = (C_p, d_p)$ be a chain complex. Assume that every chain group is decomposed $C_p = D_p \oplus E_p \oplus F_p$. The boundary maps d_p are then decomposed in 3×3 block matrices $[d_{p,i,j}]_{1 \leqslant i,j \leqslant 3}$. If every component $d_{p,2,1} : D_p \longrightarrow E_{p-1}$ is an isomorphism, then the chain complex can be canonically reduced to a chain complex (F_p, d'_p), that is, there exists a reduction $\rho = (f_p, g_p, h_p) : (C_p, d_p) \Rrightarrow (F_p, d'_p)$ where for every $p \in \mathbb{Z}$:*

$$d'_p = d_{p,3,3} - d_{p,3,1} d_{p,2,1}^{-1} d_{p,2,3}, \tag{1.10}$$

$$f_p = \begin{bmatrix} 0 & -d_{p,3,1} d_{p,2,1}^{-1} & 1 \end{bmatrix}, \; g_p = \begin{bmatrix} -d_{p,2,1}^{-1} d_{p,2,3} \\ 0 \\ 1 \end{bmatrix}, \; h_{p-1} = \begin{bmatrix} 0 & d_{p,2,1}^{-1} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

In case the chain complex C_* in Theorem 1.64 is an ACC, a reduction can be obtained by computing a *discrete vector field*, a powerful tool that has been used, for example, to describe the homotopy types of regular CW-complexes [For98].

Definition 1.65. Let $C_* = (C_p, d_p, \beta_p)$ be an ACC. A ***discrete vector field*** V on C_* is a collection of pairs $V = \{(\sigma_i, \tau_i)\}_i$ satisfying the conditions:

1. Every σ_i is a p-cell and the corresponding component τ_i is a $(p+1)$-cell. The degree p depends on i and in general is not constant.

2. Every component σ_i is a ***regular face*** of the corresponding component τ_i (it means that the coefficient of σ_i in $d\tau_i$ is 1 or -1).

3. A cell of C_* appears at most one time in V: if i is a fixed index, then $\sigma_i \neq \sigma_j$, $\sigma_i \neq \tau_j$, $\tau_i \neq \sigma_j$ and $\tau_i \neq \tau_j$ for every $j \neq i$.

The cells σ_i are ***source cells*** and the cells τ_i are ***target cells***; a cell which does not appear in V is called a ***critical cell***.

Definition 1.66. Let $C_* = (C_p, d_p, \beta_p)$ be an ACC and let $V = \{(\sigma_i, \tau_i)\}_i$ be a discrete vector field on C_*. A ***V-path*** of degree p (and length m) is a sequence $\pi = ((\sigma_{i_k}, \tau_{i_k}))_{0 \leqslant k \leqslant m}$ satisfying:

1. Every pair $(\sigma_{i_k}, \tau_{i_k})$ is a component of the vector field V and the cell τ_{i_k} is a p-cell.

2. For every $0 < k \leqslant m$, the component σ_{i_k} is a face of $\tau_{i_{k-1}}$, non necessarily regular, but different from $\sigma_{i_{k-1}}$.

Definition 1.67. Let $C_* = (C_p, d_p, \beta_p)$ be an ACC and let V be a discrete vector field on C_*. V is said to be ***admissible*** if for every $p \in \mathbb{Z}$, a function $\lambda_p : \beta_p \longrightarrow \mathbb{N}$ is provided satisfying the following property: every V-path starting from $\sigma \in \beta_p$ has a length bounded by $\lambda_p(\sigma)$.

Remark 1.68. Considering now Theorem 1.64, if $C_* = (C_p, d_p, \beta_p)$ is an ACC, each cell basis β_p can be divided by a discrete vector field V as $\beta_p = \beta_p^t + \beta_p^s + \beta_p^c$, where β_p^t, β_p^s and β_p^c are the target, source and critical cells of V, respectively. This decomposition induces a corresponding decomposition of the chain groups $C_p = C_p^t \oplus C_p^s \oplus C_p^c$, so that every differential d_p can be viewed as a 3×3 matrix. Moreover, every map $d_{p,2,1} : C_p^t \longrightarrow C_{p-1}^s$ is an isomorphism (proved in [RS10, Proposition 18]), so that this ACC satisfies the hypothesis of Theorem 1.64 and therefore the following result is obtained.

Theorem 1.69. (Vector-Field Reduction Theorem) [RS10] *Let $C_* = (C_p, d_p, \beta_p)$ be an ACC and $V = \{(\sigma_i, \tau_i)\}$ be an admissible discrete vector field on C_*. Then the vector field V defines a canonical reduction $\rho = (f_p, g_p, h_p) : (C_p, d_p) \Rrightarrow (C_p^c, d'_p)$ as in (1.10), where $C_p^c = \mathbb{Z}[\beta_p^c]$ is the free abelian group generated by the critical p-cells.*

This theorem implies in particular that the homology groups of the chain complex C_* are isomorphic to those of a smaller chain complex generated by the critical cells of the discrete vector field. This result is implemented in the Kenzo program as part of an external module working with discrete vector fields, allowing the user to determine the critical chain complex and also the maps f, g and h of the corresponding reduction. In frequent situations, if C_* is associated with a simplicial set X, $C_* = C_*(X)$, this reduction can also be used when X is considered as an ingredient in other topological constructions (such as loop spaces or twisted cartesian product) to determine their effective homology.

1.3.3 The Kenzo program

The work that will be presented in this memoir has been implemented as a new module for the computer algebra system called Kenzo. Kenzo [DRSS99] is a symbolic computation system devoted to algebraic topology. It was originally written in 1990 by Sergeraert and Rubio, under the name of EAT (Effective Algebraic Topology) [RSS97]. In 1998 it was rewritten by Sergeraert and Dousson, with the current name of Kenzo. The last official version dates from 2008, although there exists a more recent version maintained by G. Heber [Heb19] with compatibility improvements and bugfixes.

As we have said, the topological spaces considered in Kenzo are represented by means of simplicial sets (see [May67] for definitions). The main goal of Kenzo is to be able to deal with spaces of infinite nature, codified by means of the Common Lisp programming language making an intensive use of functional programming, being the only program for algebraic topology able to carry out computations over infinite structures. The program allows in particular the computation of homology and homotopy groups of complicated spaces, such as iterated loop spaces of a loop space modified by a cell attachment or components of complex Postnikov towers, which were not known before [RS06].

Homology and homotopy groups of infinite spaces are computed in Kenzo by means of the *effective homology* method, explained in Subsection 1.3.2. When an object X is built in Kenzo, an equivalence $C_*(X) \Lleftarrow\!\!\Rrightarrow E_*$ is automatically constructed, where $C_*(X)$ is the chain complex associated with X and E_* is an effective chain complex, so that the homology groups of E_* can be determined by means of elementary operations on matrices. In this way, the homology groups of the object X, which can be of infinite nature, can be obtained thanks to the isomorphism $H_*(X) \cong H_*(C_*(X)) \cong H_*(E_*)$.

Furthermore, Kenzo has been enhanced by several authors with additional modules which are available in the personal websites of the authors. In particular, there exists a module which automatizes the construction of the Whitehead tower of fibrations for the computation of homotopy groups using only one function [Her11]. In addition, there exists another module computing spectral sequences [RRS06], a useful tool of algebraic topology which in general is not constructive. This module can be applied to determine spectral sequences associated with filtered complexes of infinite type and makes it possible to determine Serre and Eilenberg–Moore spectral sequences associated with fibrations when versions with effective homology are known for the initial spaces.

1.3.3.1 Syntax in Kenzo

In order to familiarize the reader with the syntax of Kenzo and some functions that we have used in the development of this work, we present some examples explaining the construction of some Kenzo objects. Let us show how to construct *by hand* the chain complex $C_*(K)$ which generates the simplicial homology groups of the oriented simplicial complex K in Figure 1.8.

$$K = \{[a],[b],[c],[d],[e],\\ [a,b],[a,c],[a,d],\\ [b,c],[c,d],[c,e],\\ [d,e],[a,b,c]\}$$

$|K|$

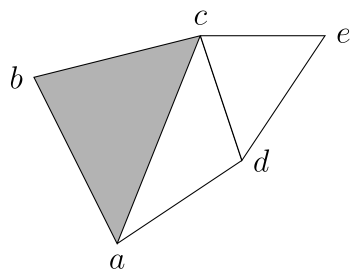


Figure 1.8: A simplicial complex K and its geometric realization $|K|$.

A chain complex in Kenzo is implemented as an instance of the class `CHAIN-COMPLEX`. To facilitate the construction of instances of this class, the software provides the function `build-chcm` defined with six keyword arguments:

- `basis` is a function defining the distinguished basis of each group C_p, or the keyword `:locally-effective` in case the chain complex is not effective.
- `cmpr` is a comparison function for the generators.
- `intr-dffr` is a Lisp function defining the differential homomorphism $d_p : C_p \longrightarrow C_{p-1}$, for each $p \in \mathbb{Z}$.
- `strt` is one of the two values: `:gnrt` or `:cmbn` defining the mapping *strategy* of the differential homomorphism.
- `bsgn` is a generator, the base point of the underlying set.
- `orgn` is a list containing a comment about the *origin* of the chain complex.

The chain complex $C_*(K)$, that we will name `example` in Kenzo, is constructed as follows:

```
> (setf example-basis #'(lambda (dmn)
                           (case dmn
                             (0 '(a b c d e))
                             (1 '(ab ac ad bc cd ce de))
                             (2 '(abc))
```

```
                              (otherwise NIL))))
> (setf example-intr-dffr
        #'(lambda (dmn gnr)
            (unless (<= 0 dmn 2)
              (error "Non-correct dimension"))
            (case dmn
              (0 (cmbn -1))
              (1 (case gnr
                   (ab (cmbn 0 -1 'a 1 'b))
                   (ac (cmbn 0 -1 'a 1 'c))
                   (ad (cmbn 0 -1 'a 1 'd))
                   (bc (cmbn 0 -1 'b 1 'c))
                   (cd (cmbn 0 -1 'c 1 'd))
                   (ce (cmbn 0 -1 'c 1 'e))
                   (de (cmbn 0 -1 'd 1 'e))))
              (2 (case gnr
                   (abc (cmbn 1 1 'ab -1 'ac 1 'bc))))
              (otherwise (error "Bad generator")))
            ))
> (setf example (build-chcm :basis example-basis
                            :cmpr #'s-cmpr
                            :intr-dffr example-intr-dffr
                            :strt :gnrt
                            :bsgn 'a
                            :orgn '(kenzo-example)))
[K1 Chain-Complex]
```

The string `[K1 Chain-Complex]` is printed by the printing method associated to the class. The identification number (1 in `K1`) is assigned automatically every time a new object is constructed in Kenzo. We can compute the homology groups of `example`:

```
> (homology example 0 3)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
Component Z
Component Z
---done---
Homology in dimension 2 :
---done---
```

Above computations show that $H_0(K) \cong \mathbb{Z}$, $H_1(K) \cong \mathbb{Z} \oplus \mathbb{Z}$ and $H_2(K) = 0$. The function `chcm-homology-gen` prints a generator for each component of the homology groups (these are represented in Figure 1.9).

```
> (chcm-homology-gen example 0)
(
----------------------------------------------{CMBN 0}
<1 * A>
----------------------------------------------
)
> (chcm-homology-gen example 1)
(
----------------------------------------------{CMBN 1}
<1 * AB>
<-1 * AD>
<1 * BC>
<1 * CE>
<-1 * DE>
----------------------------------------------

----------------------------------------------{CMBN 1}
<1 * CD>
<-1 * CE>
<1 * DE>
----------------------------------------------
)
```

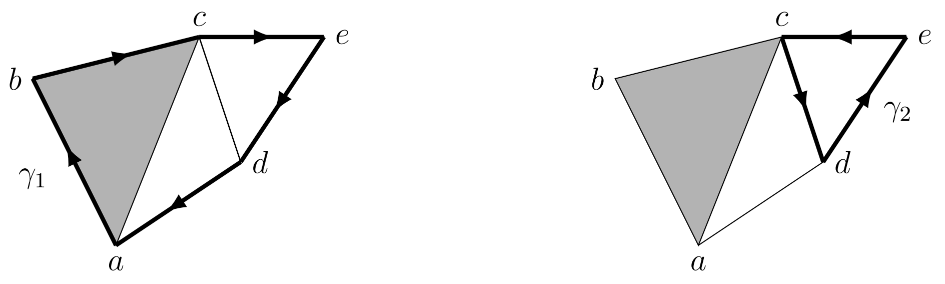


Figure 1.9: Representation of the generators of $H_1(K) = \langle\gamma_1\rangle \oplus \langle\gamma_2\rangle \cong \mathbb{Z} \oplus \mathbb{Z}$ computed in Kenzo.

The described construction of the chain complex $C_*(K)$ can be simplified by using the function `build-finite-ss`. Kenzo provides this function to create finite simplicial sets, in particular, finite simplicial complexes. Simplicial sets are implemented as an instance of the class `SIMPLICIAL-SET` which inherits from `CHAIN-COMPLEX`. In our example, we can construct K as a simplicial set in Kenzo as follows:

```
> (setf example2 (build-finite-ss '(a b c d e
                                    1 ab (b a) ac (c a)
                                      ad (d a) bc (c b)
                                      cd (d c) ce (e c)
                                      de (e d)
                                    2 abc (bc ac ab))))
Checking the 0-simplices...
Checking the 1-simplices...
Checking the 2-simplices...
[K7 Simplicial-Set]
```

Observe that `build-finite-ss` verifies the coherence of the given description of the faces (for example, regarding above computation, the faces of `bc` are those in the list `(c b)` while the faces of `abc` are in the list `(bc ac ab)`). After this verification, `build-finite-ss` calls the function `build-smst` to create an instance of the class `SIMPLICIAL-SET`, subclass of the class `CHAIN-COMPLEX`. So that, it is easy for the user to obtain the homology groups of `example2` (the same of `example`, of course):

```
> (homology example2 0 3)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
Component Z
Component Z
---done---
Homology in dimension 2 :
---done---
```

In Kenzo it has been implemented the computation of discrete vector fields on chain complexes. Regarding the space `example2`, the matrices of the differential maps d_1 and d_2 are given by

```
> (setf d1 (chcm-mat example2 1))
> (show d1)
========== MATRIX 5 row(s) + 7 column(s) ==========

-1  -1  -1   0   0   0   0
 1   0   0  -1   0   0   0
 0   1   0   1  -1  -1   0
 0   0   1   0   1   0  -1
 0   0   0   0   0   1   1

> (setf d2 (chcm-mat example2 2))
> (show d2)
```

```
========== MATRIX 7 row(s) + 1 column(s) ==========

 1
-1
 0
 1
 0
 0
 0
```

Then, we can compute two vector fields by using the matrices `d1` and `d2`:

```
> (m-vf-hard 5 (matrice-to-lmtrx d1))
((0 0) (2 1) (3 2) (4 5))
> (m-vf-hard 7 (matrice-to-lmtrx d2))
((0 0))
```

Representations of the corresponding vector fields of degrees 1 and 2 are illustrated in Figure 1.10

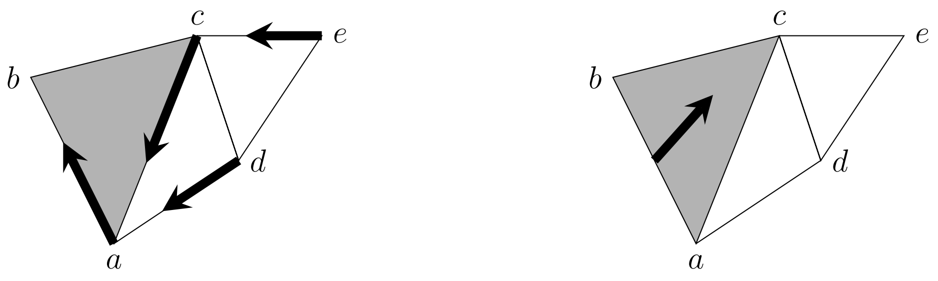


Figure 1.10: Discrete vector fields of degree 1 (left) and 2 (right) computed on `example2` in Kenzo.

Finally, let us consider an example that shows the power of Kenzo in computing homology groups of objects of infinite type. The "minimal" simplicial model of the Eilenberg–MacLane space $K(\mathbb{Z}, 1)$ (see [May67]) is defined by $K(\mathbb{Z}, 1)_n = \mathbb{Z}^n$, and it is constructed in Kenzo as follows:

```
> (setf kz1 (k-z 1))
[K16 Abelian-Simplicial-Group]
```

The Kenzo function `k-z` receives an integer n (in our example, `1`) and returns the Eilenberg–MacLane space $K(\mathbb{Z}, n)$. The abelian simplicial group $K(\mathbb{Z}, 1)$ has an infinite number of simplices in every dimension greater than or equal to 1, so that the homology groups of `kz1` cannot be elementarily computed. But the effective homology of this object, automatically computed by Kenzo and stored in the slot `efhm`, is an equivalence with the circle (1-sphere):

```
> (efhm kz1)
[K37 Homotopy-Equivalence K16 <= K16 => K31]
> (orgn (k 31))
(CIRCLE)
```

The Kenzo function `k` takes a positive integer n as input an returns the n-th object built in Kenzo. In this way, the homology groups of $K(\mathbb{Z}, 1)$ are computable through the finite equivalent object `(k 31)`:

```
> (homology kz1 0 3)
Homology in dimension 0 :
Component Z
---done---
Homology in dimension 1 :
Component Z
---done---
Homology in dimension 2 :
---done---
```

Chapter 2

A new Kenzo module for computing invariants of finite topological spaces

Despite the fact that the most important theoretical results that are the seeds of the theory of finite topological spaces were established several years ago [Ale37], [McC66], [Sto66], in recent times there has been a constant research activity about the topics involving finite spaces. Although there are computational implementations of some of the methods working on finite spaces in order to obtain topological invariants (see [Fer17b], [Ren19]), a lot of work in this area must be done to enrich and complement the developed functionalities.

In this chapter, we present a new module for the computer algebra system Kenzo for computing homology groups of finite topological spaces. This chapter is divided in four parts. In the first one, we introduce the representation of finite T_0-spaces and some basic operations of our new module such as methods of point reductions (elimination of beat point and weak points) and the construction of barycentric subdivision (some of these operations are also available in [Fer17b] and [Ren19]). In the second and third parts of the chapter we focus our attention on the wide class of *h-regular* finite spaces and we develop new algorithms which represent constructive versions of results found in [Min12] and [CO17] in order to compute homology groups and its generators. Up to our knowledge, these new Kenzo functions are the only available software to determine the homology groups of an h-regular space without constructing the order complex associated with the poset. Finally, in the last part of the chapter we study the performance of our methods and we compare them with those in [Fer17b] and [Ren19].

The work explained in this chapter has been presented in [CR18], [CRLRS18], [CR19a] and [CRLRS20a]. The programs are available at [CR20] in the folder /h-regular-homology.

2.1 Representation and basic operations of our new module in Kenzo

As seen in Subsection 1.3.3, Kenzo [DRSS99] is a symbolic computation system devoted to algebraic topology, allowing in particular to work with chain complexes and to compute homotopy and homology groups of simplicial sets. With our new module presented in this chapter, we enhance this system with new functionalities allowing the user to work with finite topological spaces and to compute their homology groups with generators and other useful information. By choosing this system, we can make use of the available functions working on matrices, chain complexes and homology groups.

2.1.1 The new class `FINITE-SPACE` in Kenzo

We have created the class `FINITE-SPACE` in our new module in Kenzo to work with finite T_0-spaces. This class is defined as follows:

```
(DEFCLASS FINITE-SPACE ()
  ((top :type matrice :initarg :top :reader top)
   (stong :type matrice :initarg :stong :reader stong)
   (heights :type list :initarg :heights :reader heights)
   (idnm :type fixnum :initform (incf *idnm-counter*)
         :reader idnm)
   (orgn :type list :initarg :orgn :reader orgn)))
```

The instances of this class represent finite T_0-spaces whose slots are described as follows:

`top` (TOPogenous matrix) is a sparse matrix of type `matrice` (the type for sparse matrices available in Kenzo) representing the topogenous matrix of the finite T_0-space. It must be upper triangular.

`stong` (STONG matrix) is a sparse matrix of type `matrice` representing the Stong matrix of the finite T_0-space. It must be upper triangular.

`heights` (HEIGHTS) is the list whose n-th element is a list containing the elements of the finite T_0-space with height equal to n.

`idnm` (IDentification NuMber) is the identification number as a Kenzo object.

`orgn` (ORiGiN) is a list with a description of the finite T_0-space.

It is imperative to set some of the slots `top` and `stong`, but they might not be specified simultaneously when a space is initialized since, if one of them is not given, it is constructed by using the other slot. `heights` is also an optional slot, which can be fixed after initialization of the space when it is necessary for specific computations. The slots `idnm` and `orgn` are included in all Kenzo objects.

The elements of a finite T_0-space in Kenzo are assumed to be natural numbers ordered in such a way that the topogenous and Stong matrices are upper triangular. We have implemented some functions for constructing finite T_0-spaces by using the class just defined:

`build-finite-space` *top strong heights orgn*
: The returned value is an instance of type `FINITE-SPACE`. This is the constructor function for the class of finite T_0-spaces. The slots are introduced by means of keywords and, as said before, not all of them must be given.

`random-finite-space` *dmns dens*
: This function constructs a random finite T_0-space of cardinality *dmns*. The parameter *dens* allows to control the density of ones in the Stong matrix of the space.

`facets-to-finite-space` *facets*
: The parameter *facets* must be a list whose elements (lists) represent the facets of a simplicial complex K with vertices indexed by natural numbers. Then, this function returns the `FINITE-SPACE` representing $\mathcal{X}(K)$, the face poset of K.

Some auxiliary functions that allow us to make more flexible the use of the class have been developed and implemented in Kenzo. They have been included in Appendix.

2.1.1.1 Constructing some finite spaces in Kenzo

Let us show how to represent a finite T_0-space in Kenzo. Consider the space X in Figure 2.1 (a). We must label the elements of the space in such a way that the corresponding Stong matrix is upper triangular. In this case, a labelling can be made from left to right and from bottom to top in the Hasse diagram of X as in Figure 2.1 (b).

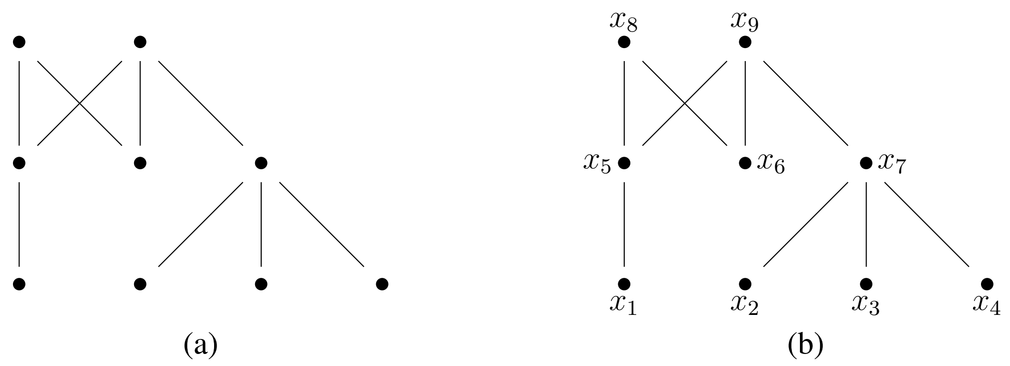


Figure 2.1: (a) Hasse diagram of a finite T_0-space X. (b) A labelling for the elements of X.

With the labelling chosen for the elements of X, the Stong matrix and the topogenous matrix are given by

$$S_X = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ & & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ & & & 1 & 0 & 0 & 1 & 0 & 0 \\ & & & & 1 & 0 & 0 & 1 & 1 \\ & & & & & 1 & 0 & 1 & 1 \\ & & & & & & 1 & 0 & 1 \\ & & & & & & & 1 & 0 \\ & & & & & & & & 1 \end{bmatrix}, \ T_X = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ & & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ & & & 1 & 0 & 0 & 1 & 0 & 1 \\ & & & & 1 & 0 & 0 & 1 & 1 \\ & & & & & 1 & 0 & 1 & 1 \\ & & & & & & 1 & 0 & 1 \\ & & & & & & & 1 & 0 \\ & & & & & & & & 1 \end{bmatrix} \qquad (2.1)$$

Observe that we can use the edges of the Hasse diagram to construct the Stong matrix in Kenzo by using the function `edges-to-stong` and constructing the `:stong` slot of a `FINITE SPACE`:

```
> (setf edges '((1 5) (2 7) (3 7) (4 7) (5 8)
                (5 9) (6 8) (6 9) (7 9)))
> (setf stongmtrx (edges-to-stong 9 edges))
> (setf finspace (build-finite-space :stong stongmtrx
                                     :orgn '(space)))
[K1 Finite-Space]
```

The slots `:top` and `:heights` were not specified for `finspace`, but the system fills them once we call them for the first time. The following auxiliar function, called `show`, allows us to plot a matrix representing a finite T_0-space.

```
> (show (top finspace))

  ========== MATRIX 9 row(s) + 9 column(s) ==========

    1   0   0   0   1   0   0   1   1
    0   1   0   0   0   0   1   0   1
    0   0   1   0   0   0   1   0   1
    0   0   0   1   0   0   1   0   1
    0   0   0   0   1   0   0   1   1
    0   0   0   0   0   1   0   1   1
    0   0   0   0   0   0   1   0   1
    0   0   0   0   0   0   0   1   0
    0   0   0   0   0   0   0   0   1

> (heights finspace)
((1 2 3 4 6) (5 7) (8 9))
```

Moreover, we have developed functions to obtain the *u-basis* $\{U_x^X\}_{x\in X}$ and the *f-basis* $\{F_x^X\}_{x\in X}$. These functions are widely used in order to compute homology invariants.

```
> (binarymatrice-to-ubasis (top finspace))
#((1) (2) (3) (4) (1 5) (6) (2 3 4 7)
  (1 5 6 8) (1 2 3 4 5 6 7 9))
> (binarymatrice-to-fbasis (top finspace))
#((1 5 8 9) (2 7 9) (3 7 9) (4 7 9)
  (5 8 9) (6 8 9) (7 9) (8) (9))
```

In order to integrate our finite topological space module in Kenzo, it is necessary to establish some links between finite spaces and Kenzo primitive structures. For instance, we can obtain a finite space from a simplicial complex.

Let us consider the simplicial complex K in Figure 2.2.

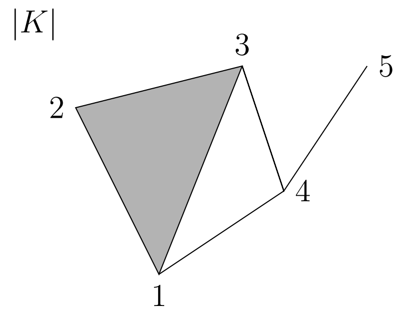


Figure 2.2: Geometric realization of a simplicial complex K.

The facets of K are the simplices $\{1,2,3\}, \{1,4\}, \{3,4\}, \{4,5\}$. Therefore, by introducing these facets as the parameter of the function `facets-to-finite-space` in Kenzo we obtain:

```
> (setf facets '((1 2 3) (1 4) (3 4) (4 5)))
((1 2 3) (1 4) (3 4) (4 5))
> (setf finspace2 (facets-to-finite-space facets))
[K2 Finite-Space]
> (stong-to-edges (stong finspace2))
((11 12) (10 12) (9 12) (5 11) (4 11) (5 10) (3 10)
 (4 9) (3 9) (5 8) (2 8) (3 7) (2 7) (2 6) (1 6))
```

In this case, `finspace2` represents the finite T_0-space $\mathcal{X}(K)$ in Figure 1.5 (note the points have been relabeled).

2.1.2 Point reductions

Once we have represented finite T_0-spaces in Kenzo, some methods have been implemented in order to identify beat points and weak points in finite spaces. With these methods, we have made functions to compute cores and *weak cores* (the term *weak core* refers to a finite T_0-space with no weak points). Some descriptions of the implemented methods and functions are given in the following lines:

`down-beat-point` *object point* `&optional` *list*
The parameter *object* could be either a topogenous matrix of type `matrice` or a `FINITE-SPACE`. The function returns a boolean deciding if *point* is a down beat point of the submatrix (or the subspace) of *object* whose indexes are in *list*. The default value of *list* is the list of all the indexes (or elements) of *object*.

`up-beat-point` *object point* `&optional` *list*
The parameter *object* could be either a topogenous matrix of type `matrice` or a `FINITE-SPACE`. The function returns a boolean deciding if *point* is an up beat point of the submatrix (or the subspace) of *object* whose indexes are in *list*. The default value of *list* is the list of all the indexes (or elements) of *object*.

`beat-point` *object point* `&optional` *list*
The parameter *object* could be either a topogenous matrix of type `matrice` or a `FINITE-SPACE`. The function returns a boolean deciding if *point* is a beat point of the submatrix (or the subspace) of *object* whose indexes are in *list*. The default value of *list* is the list of all the indexes (or elements) of *object*.

`weak-point` *object point* `&optional` *list*
The parameter *object* could be either a topogenous matrix of type `matrice` or a `FINITE-SPACE`. The function returns a boolean deciding if *point* is a weak point of the submatrix (or the subspace) of *object* whose indexes are in *list*. The default value of *list* is the list of all the indexes (or elements) of *object*.

`core-list` *object* `&optional` *list*
The parameter *object* could be either a topogenous matrix of type `matrice` or a `FINITE-SPACE`. The function returns a list of the indexes (or elements) in a core of the submatrix (or the subspace) of *object* whose indexes are in *list*. The default value of *list* is the list of all the indexes (or elements) of *object*.

`weakcore-list` *object* `&optional` *list*
The parameter *object* could be either a topogenous matrix of type `matrice` or a `FINITE-SPACE`. The function returns a list of the indexes (or elements) in a weak core of the submatrix (or the subspace) of *object* whose indexes are in *list*. The default value of *list* is the list of all the indexes (or elements) of *object*.

`core` *object* `&optional` *list*
It returns a `FINITE-SPACE` representing a core of the submatrix (or the subspace) of *object* whose indexes are in *list*.

`weakcore` *object* `&optional` *list*

It returns a `FINITE-SPACE` representing a weak core of the submatrix (or the subspace) of *object* whose indexes are in *list*.

2.1.2.1 Computing a core and a weak core of a finite space

Let us show now some examples about how our methods work in particular spaces. Consider the space in Figure 2.1, which we have represented in Kenzo by `finspace` in Subsection 2.1.1.1. Observe that a core of `finspace` is composed by the elements $\{x_5, x_6, x_8, x_9\}$:

```
> (core-list finspace)
(5 6 8 9)
```

Note that the subspace $X \smallsetminus \{x_8\}$ is contractible, since it has a maximum element, therefore its cores are one-point spaces:

```
> (core-list finspace '(1 2 3 4 5 6 7 9))
(9)
```

On the other hand, a core of the subspace $X \smallsetminus \{x_7, x_9\}$ is homeomorphic to a discrete space of 4 elements:

```
> (setf finspace3 (core finspace '(1 2 3 4 5 6 8)))
[K3 Finite-Space]
> (show (top finspace3))

========== MATRIX 4 row(s) + 4 column(s) ==========

  1   0   0   0
  0   1   0   0
  0   0   1   0
  0   0   0   1
```

Now, consider the finite T_0-space M in Figure 2.3 (a). We represent it in Kenzo by constructing `finspace4` from the tilded minimal open sets of M i.e. $\widehat{U}_x^M$. The auxiliary function `nilpot-1` fills the diagonal entries of a matrix with ones.

```
> (setf cont '(() () () (1 2) (1 2) (2 3) (2 3) (1 2 4 5)
               (1 2 3 4 6) (1 2 3 5 7) (2 3 6 7)))
> (setf tilded-ubasis (make-array 11
                            :initial-contents cont))
> (setf topogenous4 (nilpot-1 (ubasis-to-binarymatrice
                                  tilded-ubasis)))
> (setf finspace4 (build-finite-space :top topogenous4
                                                 :orgn '(space4)))
[K4 Finite-Space]
```

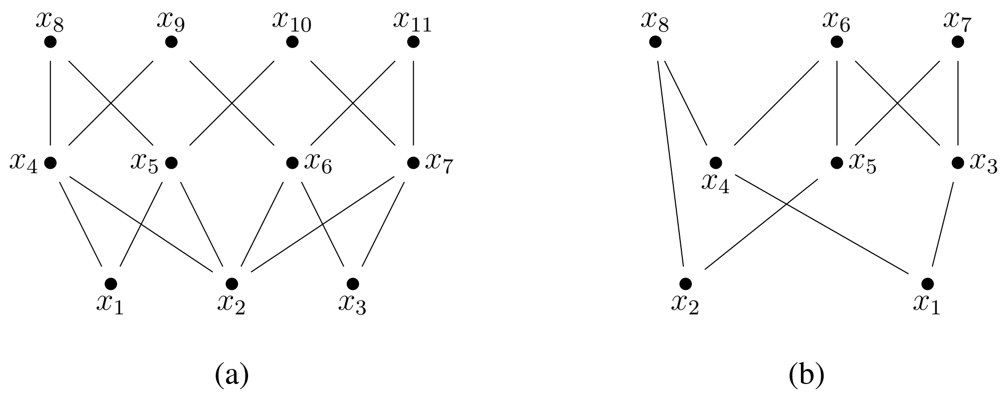


Figure 2.3: (a) Hasse diagram of the finite T_0-space M. (b) Hasse diagram of the finite T_0-space L.

Observe that M is non-contractible since it is a minimal finite space, but it is a collapsible space (x_9 is a weak point of M and $M \smallsetminus \{x_9\}$ is contractible):

```
> (core finspace4)
[K5 Finite-Space]
> (cardinality (k 5))
11
> (weakcore finspace4)
[K6 Finite-Space]
> (cardinality (k 6))
1
```

We finish these examples with a comment about the method `weakcore`. Consider the finite T_0-space L in Figure 2.3 (b) (which we represent by `finspace7` in Kenzo). We use again the function `edges-to-stong` to construct a finite space from the list of its edges.

```
> (setf edges7 '((1 3) (1 4) (2 5) (2 8) (3 6)
                 (3 7) (4 6) (4 8) (5 6) (5 7)))
> (setf stongmtrx7 (edges-to-stong 8 edges7))
> (setf finspace7 (build-finite-space :stong stongmtrx7
                                      :orgn '(space7)))
[K7 Finite-Space]
```

The method `weakcore` applied on L gives the space in Figure 2.4 (a); this procedure corresponds to the succesive collapses

$$L \searrow L \smallsetminus \{x_3\} \searrow L \smallsetminus \{x_3, x_4\} \searrow L \smallsetminus \{x_3, x_4, x_5\}.$$

On the other hand, note that x_1 is a weak point of L and `weakcore` applied to $L \smallsetminus \{x_1\}$ gives the space in Figure 2.4 (b); in this case, the represented collapses are

$$L \searrow\!\!\!\searrow L \smallsetminus \{x_1\} \searrow\!\!\!\searrow L \smallsetminus \{x_1, x_5\}.$$

```
> (weakcore finspace7)
[K8 Finite-Space]
> (cardinality (k 8))
5
> (weakcore finspace7 '(2 3 4 5 6 7 8))
[K9 Finite-Space]
> (cardinality (k 9))
6
```

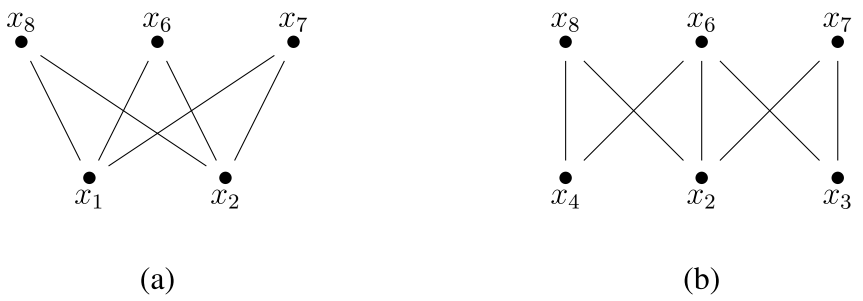


Figure 2.4: (a) `weakcore` of L. (b) `weakcore` of $L \smallsetminus \{x_1\}$.

Observe that Theorem 1.35 ensures that two cores of a finite space have the same cardinality, but the same does not occur with respect to weak points, as shown in the construction of `K8` and `K9`.

2.1.3 Computation of the barycentric subdivision

Given a finite T_0-space X, the barycentric subdivision $X' = \mathcal{X}(\mathcal{K}(X))$ and X have the same weak homotopy type. We can compute the barycentric subdivision X' of a space $X = \{x_1, \ldots, x_r\}$ by making use of its topogenous matrix T_X. More exactly, if we consider the nilpotent matrix $N_T = T_X - \mathbb{I}_r$, where $\mathbb{I}_r$ is the identity matrix of order r, we have the following result (based on ideas in [Shi68]).

Proposition 2.1. [CR16] *For each $0 \leqslant k \leqslant r-1$, the (i,j)-th entry of N_T^k represents the number of $(k+1)$-chains of X with x_i as minimum and x_j as maximum.*

A suitable adaptation of the above result allows us to calculate not only the number of n-chains but also to compute the corresponding set of n-chains of X from the successive powers of N_T, and therefore we have implemented the method

`bar-subdivision` *finspace*
which makes an instance of `FINITE-SPACE` where the `stong` slot is the Stong matrix of the barycentric subdivision of *finspace*.

The aforementioned McCord's Theorem 1.45 establishes that every finite T_0-space X is weak homotopy equivalent to its barycentric subdivision X'. In particular, the homology and homotopy groups of X can be computed by using the information given by X', since $X \overset{\text{we}}{\approx} X'$. In theory, it is a good way to describe homological and homotopical properties of X from those of X' or from its order complex $\mathcal{K}(X)$, taking advantage of geometric properties. This is the approach taken in [Fer17a] and [Ren19], where the order complex of a finite T_0-space is necessary for computing its homology.

Although Kenzo tools permit to make computations on simplicial complexes (in particular on $\mathcal{K}(X)$), the exponential growth of this complex, and therefore of X', when the size (number of points or number of edges) of X increases, limits the development of efficient programs. For this reason, we have decided to develop algorithms working directly on the space X. In the following sections we present new algorithms which are applied on the space and are valid for some particular families of finite topological spaces.

2.2 Homology of h-regular spaces

In this section we focus our attention on the particular family of h-regular finite topological spaces and we present a new algorithm implemented in Kenzo to compute their homology groups. Our algorithm provides a constructive version of Theorem 2.6, describing in an explicit way not only the groups but also the homology generators. Let us emphasize that, up to our knowledge, this new Kenzo function is the only available software to determine the homology groups of an h-regular space without constructing the order complex associated with the poset.

We begin this section by presenting some definitions and previous results which will be necessary for our work.

Definition 2.2. A ***cellular poset*** X is a graded poset such that for every $x \in X$, $\widehat{U}_x$ has the homology of the sphere in dimension $\deg(x) - 1$, where $\deg(x)$ is the degree of x in the poset. Given a cellular poset X, its ***cellular chain complex*** $C_*(X) = (C_p(X), d_p)_{p\in\mathbb{Z}}$ is defined in [Min12] by

$$C_p(X) = H_p(X^p, X^{p-1}) = \bigoplus_{\deg(x)=p} H_{p-1}(\widehat{U}_x) \tag{2.2}$$

where $H_k(Y)$ denotes the k-reduced homology group of Y and the differential $d_p : C_p(X) \longrightarrow C_{p-1}(X)$ is defined as the composition

$$H_p(X^p, X^{p-1}) \xrightarrow{\partial} H_{p-1}(X^{p-1}) \xrightarrow{j} H_{p-1}(X^{p-1}, X^{p-2})$$

where j is the canonical map induced by the inclusion and ∂ is the connecting morphism of the long exact sequence for the pair (X^p, X^{p-1}).

Note that any graded h-regular poset is cellular. The following result provides a framework to compute the homology of cellular posets.

Theorem 2.3. [Min12, Theorem 3.7] *Let X be a cellular poset and let $C_*(X)$ be its cellular chain complex. Then $H_n(C_*(X)) = H_n(X)$ for all $n \in \mathbb{N}_0$.*

The computation of homology of cellular posets described in Theorem 2.3 was generalized in [CO17] in order to be applied to a wider class of spaces by means of the notion of quasicellular space.

Definition 2.4. A finite T_0-space X is ***quasicellular*** if there exists an order preserving map $\rho : X \longrightarrow \mathbb{N}_0$, called a ***quasicellular morphism*** for X, such that

1. The set $D_p = \{x \in X : \rho(x) = p\}$ is an antichain for every $p \in \mathbb{N}_0$.

2. For every $x \in X$, the reduced homology of $\widehat{U}_x$ is concentrated in degree $\rho(x) - 1$.

Note that every cellular space is quasicellular and every h-regular space is quasicellular.

Definition 2.5. Let X be a finite T_0-space and let $\mathrm{Ch}_n(X)$ be the set of n-chains of X (chains of length n). For $s \in \mathrm{Ch}_n(X)$, the ***index*** of $y \in s$ in s is $\eta_y = \#(s \cap \widehat{U}_y)$ and the ***sign*** of y in s is defined as $\mathrm{sgn}_s(y) = (-1)^{\eta_y}$.

Bearing in mind the above definitions, the following result is stated.

Theorem 2.6. [CO17, Corollary 3.15] *Let X be a quasicellular space and let ρ be a quasicellular morphism for X. Let $C_*^\rho(X) = (C_p^\rho(X), d_p)_{p\in\mathbb{Z}}$ be the chain complex defined by*

- $C_p^\rho(X) = \bigoplus_{x\in D_p} H_{p-1}(\widehat{U}_x)$ *for each $p \in \mathbb{N}_0$ and $C_p^\rho(X) = 0$ for $p < 0$.*

- *The group homomorphism $d_p : \bigoplus_{x\in D_p} H_{p-1}(\widehat{U}_x) \longrightarrow \bigoplus_{y\in D_{p-1}} H_{p-2}(\widehat{U}_y)$, for each $p \in \mathbb{N}$, is defined by*

$$d_p\left(\left(\left[\sum_{i=1}^{l_x} a_i^x s_i^x\right]\right)_{x\in D_p}\right) = \left(\left[\sum_{x\in D_p}\sum_{s_i^x \ni y} a_i^x \mathrm{sgn}_{s_i^x}(y)(s_i^x - \{y\})\right]\right)_{y\in D_{p-1}} \tag{2.3}$$

where for every $x \in D_p$, $l_x \in \mathbb{N}$ and for every $i \in \{1, \ldots, l_x\}$, $a_i^x \in \mathbb{Z}$ and $s_i^x \in \mathrm{Ch}_{p-1}(\widehat{U}_x)$.

Then $H_n(C_*^\rho(X)) = H_n(X)$ for all $n \in \mathbb{N}_0$.

Remark 2.7. Note that if X is an h-regular space, the chain complex given in Theorem 2.6 is an ACC (recall Definition 1.59), where each basis β_p can be identified with $D_p = \{x \in X : h(x) = p\}$.

Consider an h-regular space $X = \{x_1, \ldots, x_r\}$. The map $h : X \longrightarrow \mathbb{N}_0$ such that $h(x)$ is the height of x in X, is a quasicellular morphism for X, therefore X can be regarded as a quasicellular space. Moreover, by the definition of h-regularity (Definition 1.48), $H_{p-1}(\widehat{U}_x) \cong \mathbb{Z}$ if $p = h(x)$, which means that we have to find only one generator of the $(h(x)-1)$-homology group for each $\widehat{U}_x$.

In the next lines, we are going to describe an algorithm to construct the chain complex $C^\rho_*(X) \equiv C^h_*(X)$ of Theorem 2.6, that we have implemented in the Kenzo system. The key point of the method consists in computing a matricial expression for the differential $d_n : C^h_n(X) \to C^h_{n-1}(X)$ by recursively obtaining the homology generators for the groups $\{H_{k-1}(\widehat{U}_x) : h(x) = k\}$ (from differentials and homology generators that have been previously calculated).

While we describe the construction procedure of the chain complex $C^h_*(X)$ in the following subsections, we will show the construction of this chain complex applied to the h-regular finite space in Figure 2.5.

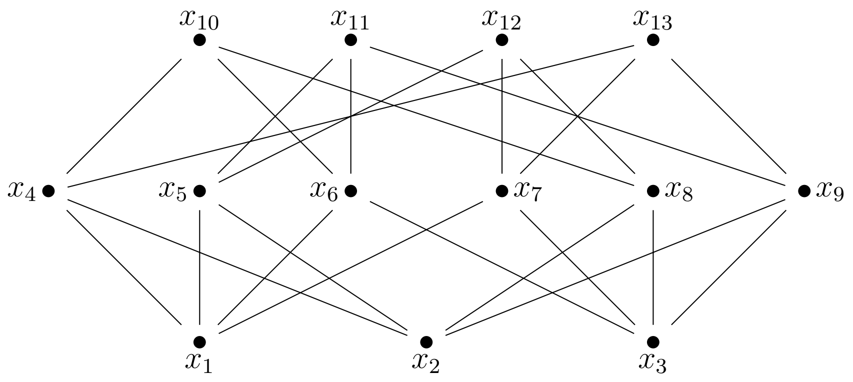


Figure 2.5: Minimal finite model for the projective plane $\mathbb{RP}^2$.

2.2.1 Computing $C^h_0(X)$ and d_0

At first, consider the construction of $C^h_0(X)$. For every $x_k \in X$ such that $h(x_k) = 0$ we have $\widehat{U}_k = \emptyset$ and in this case, we will denote the generator of $H_{-1}(\widehat{U}_k)$ by $\emptyset_k$. Therefore, if $D_0 = \{x \in X : h(x) = 0\} = \{x_{k_1}, \ldots, x_{k_{p_0}}\}$,

$$C^h_0(X) = \bigoplus_{i=1}^{p_0} \langle \emptyset_{k_i} \rangle$$

Since $C^h_{-1}(X) = 0$, the differential map $d_0 : C_0(X) = \bigoplus_{h(x)=0} \tilde{H}_{-1}(\widehat{U}_x) \longrightarrow 0$ is trivial.

In the case of Figure 2.5, the generators correspond to the elements of height 0, so that we obtain $C_0(X) = \langle x_1 \rangle \oplus \langle x_2 \rangle \oplus \langle x_3 \rangle$.

2.2.2 Computing $C_1^h(X)$ and d_1

As a second step, we construct $C_1^h(X)$. If $h(x_m) = 1$ then, by the h-regular condition, $\widehat{U}_m$ has exactly two elements whose indexes we denote by $m^{(1)}$ and $m^{(2)}$ (without loss of generality, we assume $m^{(1)} < m^{(2)}$), so that $\widehat{U}_m = \{x_{m^{(1)}}, x_{m^{(2)}}\}$. In this case, the generator of the reduced homology group in dimension 0 is the (formal) difference $\{x_{m^{(2)}}\} - \{x_{m^{(1)}}\}$, that we denote $\{x_{m^{(2)}}\}_m - \{x_{m^{(1)}}\}_m$.

Therefore if $D_1 = \{x_{m_1}, \ldots, x_{m_{p_1}}\}$,

$$C_1^h(X) = \bigoplus_{i=1}^{p_1} \left\langle \left[\{x_{m_i^{(2)}}\}_{m_i} - \{x_{m_i^{(1)}}\}_{m_i} \right] \right\rangle.$$

For $k = 1, 2$, since $\widehat{U}_{m_i^{(k)}} = \emptyset$, then the sign of $x_{m_i^{(k)}}$ in any chain is $(-1)^0 = 1$ and therefore by formula (2.3), we obtain the following expression for all $i = 1, \ldots, p_1$:

$$d_1 \left(0, \ldots, 0, \left[\{x_{m_i^{(2)}}\}_{m_i} - \{x_{m_i^{(1)}}\}_{m_i} \right], 0, \ldots, 0\right) = \emptyset_{m_i^{(2)}} - \emptyset_{m_i^{(1)}}. \tag{2.4}$$

Then, d_1 is a $p_0 \times p_1$ matrix where for all $i = 1, \ldots, p_1$, the $(m_i^{(1)}, i)$-entry equals -1 and the $(m_i^{(2)}, i)$-entry equals 1; all the other entries are equal to 0.

Regarding the space of Figure 2.5, we have that $D_1 = \{x_4, x_5, x_6, x_7, x_8, x_9\}$ and the 0-th reduced homology groups, direct summands of $C_1^h(X)$, are given by

$$\begin{array}{lcl}
\tilde{H}_0(\widehat{U}_4) = \langle \{x_2\}_4 - \{x_1\}_4 \rangle & , & \tilde{H}_0(\widehat{U}_5) = \langle \{x_2\}_5 - \{x_1\}_5 \rangle \\
\tilde{H}_0(\widehat{U}_6) = \langle \{x_3\}_6 - \{x_1\}_6 \rangle & , & \tilde{H}_0(\widehat{U}_7) = \langle \{x_3\}_7 - \{x_1\}_7 \rangle \\
\tilde{H}_0(\widehat{U}_8) = \langle \{x_3\}_8 - \{x_2\}_8 \rangle & , & \tilde{H}_0(\widehat{U}_9) = \langle \{x_3\}_9 - \{x_2\}_9 \rangle
\end{array}$$

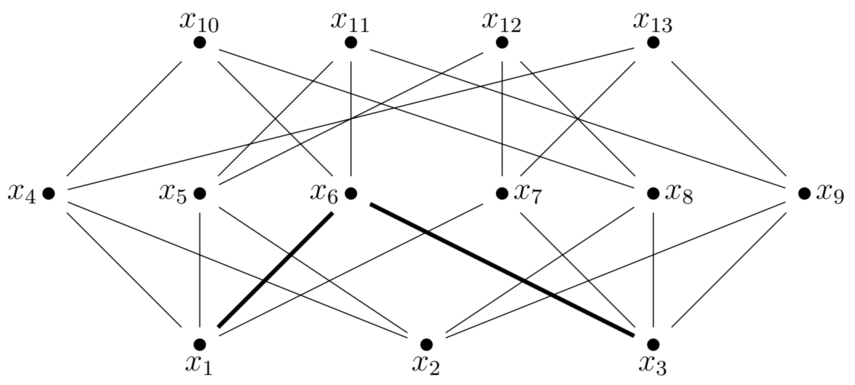


Figure 2.6: $\{x_3\}_6 - \{x_1\}_6$ is the generator of the 0-th reduced homology of $\widehat{U}_6$. Note that, by using (2.4), $d_1\,(0, 0, \{x_3\}_6 - \{x_1\}_6, 0, 0, 0) = \emptyset_3 - \emptyset_1$.

Note that if $d_1 : C_1(X) = \bigoplus_{x\in D_1} \tilde{H}_0(\widehat{U}_x) \longrightarrow \langle\emptyset_1\rangle \oplus \langle\emptyset_2\rangle \oplus \langle\emptyset_3\rangle$, by means of the expresion in (2.4), its matricial representation is

$$d_1 = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2.5}$$

2.2.3 Computing $C_n^h(X)$ and d_n, for $n \geqslant 2$

Suppose now that we have computed a generator of $H_{k-1}(\widehat{U}_y)$ for each point y such that $h(y) = k$ (for $k < n$). Now, if we consider $x_t \in X$ with $h(x_t) = n$, we have to find a generator of $H_{n-1}(\widehat{U}_t)$, that we denote by $\widehat{x}_t$. Since the height of $\widehat{U}_t$ is $n-1$, the chain complex $C_*^h(\widehat{U}_t)$ is

$$\cdots \longrightarrow 0 \longrightarrow C_{n-1}^h(\widehat{U}_t) \xrightarrow{d_{n-1}^*} C_{n-2}^h(\widehat{U}_t) \longrightarrow \cdots$$

where d_{n-1}^* represents the restriction of d_{n-1} to $\widehat{U}_t$.

Then, it suffices to compute a generator of $\ker(d_{n-1}^*)$. In this way, we obtain $\widehat{x}_t$ as a linear combination of those $\widehat{x}_y$ such that $h(y) = n-1$ and $y < x_t$. If $D_{n-1} = (y_{u_1}, \dots, y_{u_{p_{n-1}}})$, we extend the coefficients of $\widehat{x}_t$ to a vector $(A_t^1, A_t^2, \dots, A_t^{p_{n-1}})$ where $A_t^i = 0$ if $y_{u_i} \notin \widehat{U}_t$.

Now, if $D_n = \{x_{t_1}, \dots, x_{t_{p_n}}\}$,

$$C_n^h(X) = \bigoplus_{j=1}^{p_n} \langle \widehat{x}_{t_j} \rangle.$$

Regarding to the differential $d_n : C_n^h(X) \to C_{n-1}^h(X)$, it is not difficult to deduce from Theorem 2.6 that

$$d_n(\widehat{x}_t) = (-1)^{n-1}(A_t^1, A_t^2, \dots, A_t^{p_{n-1}}) \begin{pmatrix} \widehat{y}_{u_1} \\ \vdots \\ \widehat{y}_{u_{p_{n-1}}} \end{pmatrix}.$$

So, the matricial expression of d_n is directly given by the expression of the generators of dimension $n-1$. Moreover, the method allows us to describe each generator $\widehat{x}_t$ as a linear combination of $(n-1)$-chains in $\widehat{U}_t$.

Turning to the space of Figure 2.5, we have that $D_2 = \{x_{10}, x_{11}, x_{12}, x_{13}\}$. For example, in order to find the reduced homology group $\tilde{H}_1(\widehat{U}_{10})$, first we need to compute the kernel of $d_1|_{\widehat{U}_{10}}$:

$$\ker(d_1|_{\widehat{U}_{10}}) = \ker \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 1 \end{bmatrix} = \langle [-1, 1, -1]^T \rangle.$$

The vector components of the kernel generator allows us to obtain the generator of the group $\tilde{H}_1(\widehat{U}_{10})$:

$$\widehat{x}_{10} = -\widehat{x}_4 + \widehat{x}_6 - \widehat{x}_8 = -(\{x_2\}_4 - \{x_1\}_4) + (\{x_3\}_6 - \{x_1\}_6) - (\{x_3\}_8 - \{x_2\}_8).$$

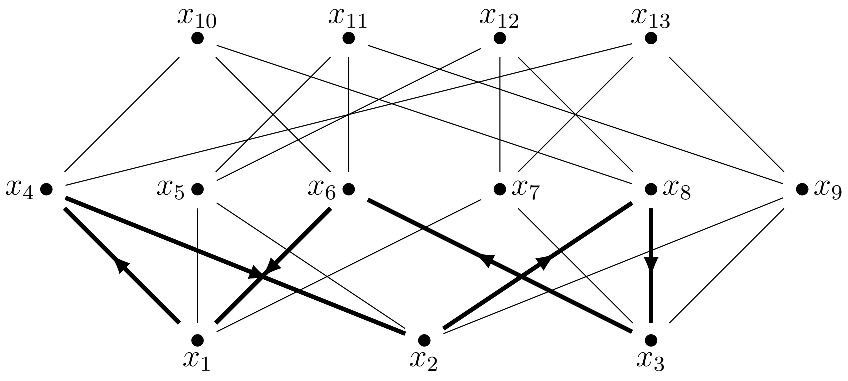


Figure 2.7: The generator $\widehat{x}_{10}$ seen as the cycle in the Hasse diagram.

In order to construct the matrix representing the differential map d_2, observe that, since $\widehat{U}_{10} \cap D_1 = \{x_4, x_6, x_8\}$, the vector generator of the above kernel must be extended to a vector of dimension $\#D_1 = 6$ by completing with zeros the positions corresponding to x_5, x_7 and x_9 (the elements in $D_1 \setminus \widehat{U}_{10}$). In this way, we obtain the first column of d_2 corresponding to x_{10}. The same process is applied to determine all columns.

$$d_2 = \begin{bmatrix} -1 & 0 & 0 & -1 \\ 0 & -1 & -1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 \end{bmatrix}. \tag{2.6}$$

2.2.4 Implementation in Kenzo

In this section, we present our new Kenzo functions for computing homology groups of h-regular spaces and we develop some examples showing how the algorithms described work on different finite spaces.

2.2.4.1 Implemented functions for computing homology

We have implemented some methods and functions in Kenzo in order to compute the homology groups and its generators of h-regular finite spaces. These new methods have been combined with Kenzo tools to compute the homology of chain complexes.

`h-regular-dif` *finspace*
: This method takes an h-regular finite space X of type `FINITE-SPACE` as the input parameter *finspace* and returns a list whose n-th element is the matrix of the differential map d_n of the complex $C^h_*(X)$.

`chcm-h-regular` *finspace*
: This function returns a Kenzo `CHAIN-COMPLEX` representing the complex $C^h_*(X)$ described in above subsections, by using the matrices in the output list of the function `h-regular-dif` applied to *finspace*.

`h-regular-homology` *finspace dim*
: It computes the homology group of *finspace* in dimension *dim* by means of $C^h_*(X)$.

`h-regular-homology-generators` *finspace dim*
: It returns the generators of the homology group (`h-regular-homology` *finspace dim*).

`explicit-chcm-h-regular` *finspace*
: This function returns a Kenzo `CHAIN-COMPLEX` representing the complex $C^h_*(X)$ described in above subsections, by using the matrices in the output list of the function `h-regular-dif` applied to *finspace*. In this case, the generators are seen as chains in the Hasse diagram of the h-regular space.

`explicit-h-regular-homology-generators` *finspace dim*
: It returns an expression of the generators (`h-regular-homology-generators` *finspace dim*) as linear combinations of edges of the Hasse diagram of the h-regular space.

Some examples of computations with these functions are explained below.

2.2.4.2 A didactical example

Consider the finite space whose Hasse diagram is shown in Figure 2.8. Note that this space is h-regular, it has no weak points and it is not the face poset of a regular CW-complex. This poset, that we have called `finite-space`, can be represented in Kenzo by defining an array of lists of the elements covered by each point of the space as follows:

```
> (setf cont '(() () () () (1 2) (1 2) (2 3)
              (2 4) (2 3) (2 4) (3 4) (3 4)
              (5 6) (5 6 7) (5 6 7 8) (7 8 11)
              (8 9 12) (9 10 11) (7 10 12)))
> (setf edges (make-array 19 :initial-contents cont))
> (setf finite-space (build-finite-space
                         :stong (edges-to-stong-mtrx edges)
                         :orgn `(example)))
[K1 Finite-Space]
```

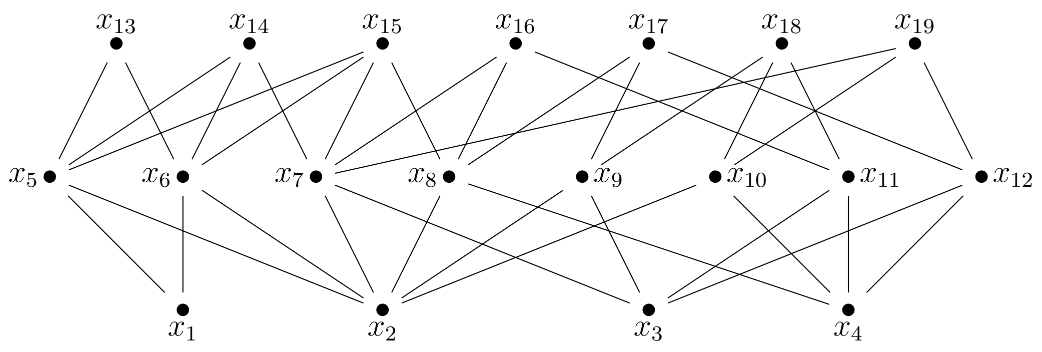


Figure 2.8: Hasse diagram of `finite-space`.

Let us show two alternatives to compute a chain complex which generates the homology of `finite-space` and compare them: the first one is to compute its barycentric subdivision and to apply the methods of above subsections on it (remember that the barycentric subdivision of a finite space is an h-regular space) and the second one is to apply such methods directly on `finite-space`.

Computing the barycentric subdivision

At first, we compute the barycentric subdivision of `finite-space`, which we call `sd-finite-space` in Kenzo, and its chain complex `chcm-h-regular`:

```
> (setf sd-finite-space (bar-subdivision finite-space))
[K2 Finite-Space]
> (setf chcm-sd (chcm-h-regular sd-finite-space))
[K3 Chain-Complex]
```

In this case we have the following number of generators in dimensions 0, 1 and 2:

```
> (length (basis chcm-sd 0))
19
> (length (basis chcm-sd 1))
58
> (length (basis chcm-sd 2))
42
```

Applying the methods directly

Since `finite-space` is an h-regular space we can obtain the chain complex of Theorem 2.6 as follows:

```
> (setf chcm (chcm-h-regular finite-space))
```

```
[K5 Chain-Complex]
```

Remember that in this case, the number of generators of dimension p corresponds to the number of elements of height p in the space (see Figure 2.8):

```
> (length (basis chcm 0))
4
> (length (basis chcm 1))
8
> (length (basis chcm 2))
7
```

Then, for example, if we require the generator (of $C_2^h(X)$ in Theorem 2.6) associated to the element x_{16} in Figure 2.8 in terms of `chcm` we get:

```
> (fourth (basis chcm 2))
X16
```

Even more, if we want to know explicitly this generator (in terms of chains in the space), we require the chain complex `explicit-chcm-h-regular` to Kenzo in order to obtain the following expression:

```
> (setf explicit-chcm (explicit-chcm-h-regular finite-space))
[K7 Chain-Complex]
> (fourth (basis explicit-chcm 2))
----------------------------------------------{CMBN 2}
<1 * (3 11 16)>
<-1 * (4 11 16)>
<-1 * (2 8 16)>
<1 * (4 8 16)>
<1 * (2 7 16)>
<-1 * (3 7 16)>
----------------------------------------------
```

This is the generator of the homology group $H_1(\widehat{U}_{16})$, so that the output has to be interpreted in the poset as a formal sum of chains:

$$\{x_3, x_{11}\} - \{x_4, x_{11}\} - \{x_2, x_8\} + \{x_4, x_8\} + \{x_2, x_7\} - \{x_3, x_7\}.$$

Observe that each term of the output has the number 16 as last element, representing the dependence of the above sum to the generator of x_{16}; this is important to be able to distinguish between, for example, the generators of $H_0(\widehat{U}_5)$ and $H_0(\widehat{U}_6)$ whose interpretation as formal sum of chains is $\{x_2\} - \{x_1\}$ in both cases.

Of course the returned results regarding to homology are the same as those obtained for the barycentric subdivision `sd-finite-space`, but in this case we can also compute the generators of such homology groups and interpret them as formal chains on the space. For example, the generator of the homology group in dimension 1 is given by:

```
> (h-regular-homology-generators finite-space 1)
(
----------------------------------------------{CMBN 1}
<1 * X8>
<-1 * X10>
----------------------------------------------
)
> (explicit-h-regular-homology-generators finite-space 1)
(
----------------------------------------------{CMBN 1}
<1 *
----------------------------------------------{CMBN 1}
<-1 * (2 8)>
<1 * (4 8)>
----------------------------------------------
>
<-1 *
----------------------------------------------{CMBN 1}
<-1 * (2 10)>
<1 * (4 10)>
----------------------------------------------
>
----------------------------------------------
)
```

which represents the cycle $-\{x_2, x_8\} + \{x_4, x_8\} + \{x_2, x_{10}\} - \{x_4, x_{10}\}$ in Figure 2.8.

2.3 Using discrete vector fields on finite topological spaces

Discrete Morse Theory was introduced by Robin Forman in 1998 [For98]. One of the main goals of this theory is to describe the relation between the singularities (critical points) of a discrete Morse function and the topology of the object over such function is defined (triangulations of varieties or, more generally, simplicial complexes). As in classical Morse theory [Mil63], it is of special interest to determine the functions with the least possible number of critical points, a problem that is closely related to the homology and homotopy of the triangulation considered. On the other hand, taking into account the combinatorial nature of Discrete Morse Theory, it can be interpreted from the point of view of graph theory in terms of matchings on the Hasse diagrams of the face posets of the simplicial complexes considered.

In [Min12], the Discrete Morse Theory for the class of h-regular posets is introduced, extending the main results of the theory to this wider class. In this section, we will describe an algorithm to compute suitable matchings on the Hasse diagrams of h-regular posets in order to give a constructive version of Theorem 2.10. Then, by using this matching and the relation with the ACC given in Remark 2.7, we can use the decomposition in Remark 1.68 in order to find a chain complex, depending on the critical points respect to the computed matching, which allows to obtain the homology of h-regular finite spaces. Implementation of our algorithms in the Kenzo system and examples of computations by using them will also be shown.

We begin the section with some definitions and results which will be used in our work.

Definition 2.8. Let X be a poset. An edge $(x, y) \in \mathrm{E}(\mathcal{H}(X))$ is ***homologically admissible*** if the subposet $\widehat{U}_y - \{x\}$ is acyclic. A poset is ***homologically admissible*** if all its edges are homologically admissible.

It can be proved that any homologically admissible poset is cellular. The face poset $\mathcal{X}(K)$ of any regular CW-complex K (in particular, of any finite simplicial complex) is homologically admissible [Min12, Remarks 2.6 and 3.10].

Definition 2.9. Let X be a finite T_0-space and let $\mathcal{H}(X)$ be its Hasse diagram. A ***matching*** M on $\mathcal{H}(X)$ is a set of edges where no edges share a common vertex. A matching M on $\mathcal{H}(X)$ is a ***Morse matching*** provided that the directed graph $\mathcal{H}(X)$, modified by reversing the orientation of the edges in M, is acyclic. M is called ***homologically admissible*** if all its edges are homologically admissible. A point of X is called ***critical*** (respect to M) if it is not incident to any edge in M.

The following result asserts that the homology of a quasicellular space X coincides with the homology of a complex $C^c_*(X)$, obtained by restricting only to those direct summands of (2.3) corresponding to the set of critical points of a homologically admissible Morse matching M.

Theorem 2.10. [CO17, Theorem 4.3] *Let X be a quasicellular poset and let M be a homologically admissible Morse matching on $\mathcal{H}(X)$. For $p \in \mathbb{N}_0$, let $A_p = \{x \in D_p :$ x is a critical point of $X\}$. Then $H_n(C^c_*(X)) = H_n(X)$ for all $n \in \mathbb{N}_0$, where $C^c_*(X) = (C^c_p(X), d'_p)_{p\in\mathbb{Z}}$ is defined by*

$$C^c_p(X) = \bigoplus_{x\in A_p} H_{p-1}(\widehat{U}_x) \tag{2.7}$$

(the differentials d'_p can be theoretically defined from those of (2.3)).

2.3.1 Constructing a homologically admissible Morse matching

Note that Theorem 1.69 provides an explicit (algebraic) description of the reduction induced by an admissible vector field on a chain complex. Now, we are interested in applying this kind of reduction to our construction of the chain complex $C_*^h(X)$ in Theorem 2.6. This will allow us to obtain a representation in Kenzo of the reduced chain complex $C_*^c(X)$ defined in Theorem 2.10. In order to do this, we are going to define the corresponding notion of homologically admissible Morse matching on a finite T_0-space X.

In [RS10], an algorithm is presented for computing an admissible vector field for an ACC of finite type. This algorithm is based on the idea of vector field for a matrix M, which can be applied on the differential matrices of the complex.

Definition 2.11. A ***vector field*** V for a matrix M with m rows and n columns is a set of integer pairs $\{(a_i, b_i)\}_i$ satisfying three conditions:

1. $1 \leqslant a_i \leqslant m$ and $1 \leqslant b_i \leqslant n$.
2. The (a_i, b_i)-entry of the matrix is ± 1.
3. The indexes a_i (resp. b_i) are pairwise different.

The aforementioned algorithm makes use of the following result: a vector field V is admissible if and only if there is no loop $a_1 > a_2 > \cdots > a_k = a_1$, where $>$ is the partial order between source cells defined by: $a > a'$ if and only if a V-path goes from a to a'.

The idea of the algorithm consists in trying to add a new vector to an admissible vector field already constructed, in such a way the new vector field keeps the admissibility property. The process starts from the void vector field and, by running the successive rows of the matrix M in the usual reading order, the algorithm checks if a candidate to vector keeps admissibility; if this is the case, the vector is added to the vector field and the partial order is uploaded. For example, in [RS10], by using this algorithm, the vector field $V_M = \{(1,3), (2,2), (3,4), (5,1)\}$ is obtained for the matrix

$$M = \begin{bmatrix} 0 & 0 & \boxed{-1} & -1 & 0 \\ 0 & \boxed{-1} & 0 & 0 & 1 \\ 0 & 0 & 0 & \boxed{1} & 1 \\ 0 & -1 & 1 & 0 & -1 \\ \boxed{-1} & 1 & -1 & 0 & 0 \end{bmatrix}.$$

Supposing that M represents the differential map of an ACC in degree p, $d_p : C_p = \langle g_1, \ldots, g_5 \rangle \to C_{p-1} = \langle g'_1, \ldots, g'_5 \rangle$, then the computed vector field V_M corresponds to the list $\{(g'_1, g_3), (g'_2, g_2), (g'_3, g_4), (g'_5, g_1)\}$. Computing the reduced chain complex of Theorem 1.69 and repeating the process for different degrees, new vectors can be added to the previous list, obtaining an admissible vector field for the corresponding ACC.

Now, we can try to adapt the above procedure to obtain a Morse matching on the Stong matrix of a finite h-regular T_0-space X; we can not do it directly because it is necessary, by the definition of a matching, to avoid pairs with repeated components (like $(2, 2)$ in V_M) and different pairs with components in common (like $(1, 3)$ and $(5, 1)$ in V_M). Therefore, avoiding the problems just mentioned and inspired on the above algorithm, we have developed a new one which computes a Morse matching directly on the Stong matrix of X.

On the other hand, let $(x, y) \in \mathrm{E}(\mathcal{H}(X))$. If we verify the contractibility of $\widehat{U}_y - \{x\}$, the edge (x, y) will be homologically admissible (Definition 2.8); since a finite topological space is contractible if and only if its core reduces to a single point, we can check the homological admissibility of an edge (x, y) by using the method `core` applied to $\widehat{U}_y - \{x\}$. This verification is implemented in the new algorithm in order to check if a candidate to vector represents a homologically admissible edge.

Remark 2.12. Let us note that, if the edge (x, y) is homologically admissible, then the coefficient of the generator $\widehat{y}$ in $\widehat{x}$ is equal to ± 1. Then, the method ensures that a discrete vector field on the chain complex $C_*^h(X)$ is obtained.

2.3.1.1 Example

Let us show how the algorithm for computing a Morse matching is applied to the finite h-regular T_0-space X in Figure 2.5. Note that this space is homologically admissible. The (upper triangular) Stong matrix of X is

$$S_X = \begin{bmatrix}
1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 & & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 & & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
 & & & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
 & & & & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 & & & & & & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
 & & & & & & & 1 & 0 & 1 & 0 & 1 & 0 \\
 & & & & & & & & 1 & 0 & 1 & 0 & 1 \\
 & & & & & & & & & 1 & 0 & 0 & 0 \\
 & & & & & & & & & & 1 & 0 & 0 \\
 & & & & & & & & & & & 1 & 0 \\
 & & & & & & & & & & & & 1
\end{bmatrix}.$$

The algorithm starts with $M = \{\}$, the empty set. We are going to add edges to M in order to obtain a homologically admissible Morse matching on $\mathcal{H}(X)$. We search on the columns of S_X (equivalent to search by row) the first nonzero entry (from top to bottom) not in the diagonal of the matrix. Note that the first three columns have only nonzero entries on the diagonal, but in fourth column, the entry $(1, 4)$ is the first 1 found; this entry corresponds to the edge (x_1, x_4) which we use to update: $M = \{(x_1, x_4)\}$.

Once we have added to M the edge corresponding to entry $(1, 4)$, we will not search ones in rows and columns with indexes 1 and 4. In order to remember the used indexes, we consider the set $I = \{1, 4\}$. Moreover, when an edge is put in M, the order of the involved elements is reversed (following Definition 2.9) and we register such data in the set $R = \{x_4 < x_1\}$.

In what follows, a box $\boxed{1}$ in S_X means that the corresponding edge was already included in M and a circle $\textcircled{1}$ means that we are studying if such edge could be part of M or not.

Now, we continue searching in column 5 nonzero entries. Note that entry $(1,5)$ of S_X can not be considered since $1 \in I$. This is not the case with the entry $(2,5)$, whose indexes do not belong to I.

$$S_X = \begin{bmatrix} 1 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 1 & \textcircled{1} & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ & & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ & & & & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ & & & & & & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ & & & & & & & 1 & 0 & 1 & 0 & 1 & 0 \\ & & & & & & & & 1 & 0 & 1 & 0 & 1 \\ & & & & & & & & & 1 & 0 & 0 & 0 \\ & & & & & & & & & & 1 & 0 & 0 \\ & & & & & & & & & & & 1 & 0 \\ & & & & & & & & & & & & 1 \end{bmatrix}.$$

However, if the edge (x_2, x_5) would be part of M, we would have to reverse its direction and the relation $x_5 < x_2$ would be included in R, but this is not possible since in such case, we would have a cycle $x_1 < x_5 < x_2 < x_4 < x_1$, which contradicts the definition of Morse matching.

In column 6, the entry $(3,6)$ has no indexes in I and it does not generate a cycle in the modified Hasse diagram of X so that we update the sets:

$$\begin{aligned} M &= \{(x_1,x_4),(x_3,x_6)\}\ , \\ I &= \{1,3,4,6\}\ , \\ R &= \{x_4 < x_1, x_6 < x_3\}. \end{aligned}$$

Note that in column 7 there are no available entries, since the indexes of all the nonzero entries not in the diagonal are in I. In column 8, the entry $(2,8)$ has no indexes in I, but if we reverse the direction of such edge, we obtain the cycle $x_8 < x_2 < x_4 < x_1 < x_6 < x_3 < x_8$. In column 9, the only available entry is $(2,9)$, but by reversing the direction of the corresponding edge, we have the cycle $x_9 < x_2 < x_4 < x_1 < x_6 < x_3 < x_9$.

In column 10, entries $(4,10)$ and $(6,10)$ have row index in I, but $(8,10)$ has no indexes in I and it does not generate a cycle, allowing to update the considered sets:

$$\begin{aligned} M &= \{(x_1,x_4),(x_3,x_6),(x_8,x_{10})\}\ , \\ I &= \{1,3,4,6,8,10\}\ , \\ R &= \{x_4 < x_1, x_6 < x_3, x_{10} < x_8\}. \end{aligned}$$

In column 11, the entry $(5,11)$ allows to update the sets and the same occur with the entry $(7,12)$ in column 12:

$$\begin{aligned} M &= \{(x_1,x_4),(x_3,x_6),(x_8,x_{10}),(x_5,x_{11}),(x_7,x_{12})\}\ , \\ I &= \{1,3,4,5,6,7,8,10,11,12\}\ , \\ R &= \{x_4 < x_1, x_6 < x_3, x_{10} < x_8, x_{11} < x_5, x_{12} < x_7\}. \end{aligned} \tag{2.8}$$

Finally, in column 13 the only available entry with no indexes in I is $(9,13)$.

$$S_X = \begin{bmatrix} 1 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ & & 1 & 0 & 0 & \boxed{1} & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ & & & & 1 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 1 & 0 \\ & & & & & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ & & & & & & 1 & 0 & 0 & 0 & 0 & \boxed{1} & 1 \\ & & & & & & & 1 & 0 & \boxed{1} & 0 & 1 & 0 \\ & & & & & & & & 1 & 0 & 1 & 0 & \textcircled{1} \\ & & & & & & & & & 1 & 0 & 0 & 0 \\ & & & & & & & & & & 1 & 0 & 0 \\ & & & & & & & & & & & 1 & 0 \\ & & & & & & & & & & & & 1 \end{bmatrix}.$$

Note that this entry can not be included in the Morse matching, since it would generate the cycle $x_{13} < x_9 < x_{11} < x_5 < x_{12} < x_7 < x_{13}$, using the reversed relations in R. At this point, the algorithm ends the searching, obtaining the homologically admissible Morse matching M shown in (2.8), which is represented in Figure 2.9.

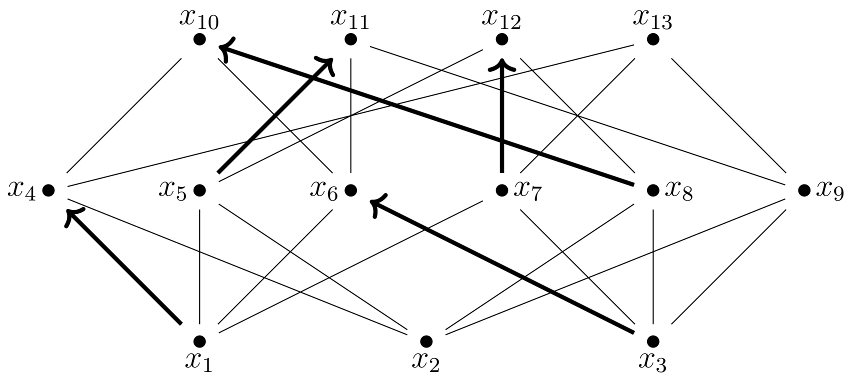


Figure 2.9: A Morse matching on the h-regular space of Figure 2.5.

2.3.2 Constructing the chain complex restricted to critical cells

Having constructed a homologically admissible Morse matching on $\mathcal{H}(X)$, we use the decomposition described in Remark 1.68 to write the matrix representing the differential map d_p of the chain complex of Remark 2.7 as follows:

$$d_p = \begin{array}{c|c|c|c|} & \beta_p^t & \beta_p^s & \beta_p^c \\ \hline \beta_{p-1}^t & d_{p,1,1} & d_{p,1,2} & d_{p,1,3} \\ \hline \beta_{p-1}^s & d_{p,2,1} & d_{p,2,2} & d_{p,2,3} \\ \hline \beta_{p-1}^c & d_{p,3,1} & d_{p,3,2} & d_{p,3,3} \\ \hline \end{array} \tag{2.9}$$

Remark 2.13. Observe that $d_{p,2,1}$ is a square block. Let us assume that the order of the elements of the bases β_{p-1}^s and β_p^t is such that the diagonal of $d_{p,2,1}$ is composed by 1's and -1's (this is justified by 2. in Definition 1.65).

In order to compute d'_p in (1.10), we consider the submatrix of d_p formed by the blocks $d_{p,2,1}$, $d_{p,2,3}$, $d_{p,3,1}$ and $d_{p,3,3}$; then we apply elementary column operations on such a submatrix, transforming $d_{p,2,1}$ to an identity matrix $\mathbb{I}$ and modifying the block $d_{p,3,1}$:

$$\begin{array}{|c|c|} \hline d_{p,2,1} & d_{p,2,3} \\ \hline d_{p,3,1} & d_{p,3,3} \\ \hline \end{array} \longrightarrow \begin{array}{|c|c|} \hline \mathbb{I} & d_{p,2,3} \\ \hline d^*_{p,3,1} & d_{p,3,3} \\ \hline \end{array}. \tag{2.10}$$

Since the operations applied to the block $d_{p,2,1}$ in order to obtain the identity matrix $\mathbb{I}$ correspond to the inverse matrix $d_{p,2,1}^{-1}$, one has $d^*_{p,3,1} = d_{p,3,1}d_{p,2,1}^{-1}$, and applying Theorem 1.64 one obtains $d'_p = d_{p,3,3} - d_{p,3,1}d_{p,2,1}^{-1}d_{p,2,3} = d_{p,3,3} - d^*_{p,3,1}d_{p,2,3}$.

Regarding the homologically admissible Morse matching shown in Figure 2.9, the corresponding decomposition described in Remark 1.68 is given by

$$\begin{array}{lll} \beta_0^t = \emptyset & \beta_1^t = \{x_4, x_6\} & \beta_2^t = \{x_{10}, x_{11}, x_{12}\} \\ \beta_0^s = \{x_1, x_3\} & \beta_1^s = \{x_8, x_5, x_7\} & \beta_2^s = \emptyset \\ \beta_0^c = \{x_2\} & \beta_1^c = \{x_9\} & \beta_2^c = \{x_{13}\} \end{array}$$

Note that in β_1^s, we have chosen the order x_8, x_5, x_7, which ensures the mentioned property in Remark 2.13; the *natural* order x_5, x_7, x_8 allows zero entries in the diagonal of the block $d_{2,1}$, the reason why we did not choose it.

Bearing in mind the above decomposition, the matrices d_1 and d_2 in (2.5) and (2.6) can be written as in (2.9):

$$d_1 = \left[\begin{array}{cc|ccc|c} \hline -1 & -1 & 0 & -1 & -1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ \hline 1 & 0 & -1 & 1 & 0 & -1 \end{array}\right] \quad , \quad d_2 = \left[\begin{array}{ccc||c} -1 & 0 & 0 & -1 \\ 1 & 1 & 0 & 0 \\ \hline -1 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 \\ \hline 0 & -1 & 0 & -1 \end{array}\right].$$

The submatrices formed by the blocks $d_{2,1}$, $d_{2,3}$, $d_{3,1}$ and $d_{3,3}$ of each matrix are the following:

$$\left[\begin{array}{cc|c} -1 & -1 & 0 \\ 0 & 1 & 1 \\ \hline 1 & 0 & -1 \end{array}\right] \quad , \quad \left[\begin{array}{ccc|c} -1 & 0 & -1 & 0 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 \\ \hline 0 & -1 & 0 & -1 \end{array}\right].$$

Performing elementary column operations on above submatrices in order to transform $d_{2,1}$ blocks to identity matrices, we obtain:

$$\left[\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 1 \\ \hline -1 & -1 & -1 \end{array}\right] \quad , \quad \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \hline 0 & 1 & 1 & -1 \end{array}\right].$$

Therefore, by using the relation $d'_p = d_{p,3,3} - d_{p,3,1}d^{-1}_{p,2,1}d_{p,2,3}$, we obtain the differential maps d'_p of the chain complex $C^c_*(X)$ in Theorem 2.10:

$$d'_1 = \begin{bmatrix} -1 \end{bmatrix} - \begin{bmatrix} -1 & -1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix},$$

$$d'_2 = \begin{bmatrix} -1 \end{bmatrix} - \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \end{bmatrix}.$$

Hence, the chain complex $C^c_*(X)$

$$\cdots \longrightarrow 0 \longrightarrow \langle \widehat{x}_{13} \rangle \xrightarrow{[-2]} \langle \widehat{x}_9 \rangle \longrightarrow \langle \widehat{x}_2 \rangle \longrightarrow 0 \longrightarrow \cdots$$

allows us to compute the non-trivial homology groups of the h-regular space X in Figure 2.5: $H_0(X) \cong \mathbb{Z}$ and $H_1(X) \cong \mathbb{Z}/2\mathbb{Z}$.

2.3.3 Implementation in Kenzo

In this subsection, we present our new Kenzo functions for computing discrete vector fields on h-regular spaces. Moreover, we use them to compute homology of the same examples presented in Subsection 2.2.4.2.

2.3.3.1 Implemented functions for working with discrete vector fields and homology

We have designed some functions to compute a homologically admissible Morse matching on a finite space as well as functions to obtain the homology groups and their generators of an h-regular finite space in Kenzo. Some descriptions are given in the next lines.

`dvfield` *finspace*
: It computes a homologically admissible Morse matching on the finite space *finspace*.

`h-regular-dif-dvf` *finspace targets sources*
: This method takes an h-regular finite space X of type `FINITE-SPACE` as the input parameter *finspace* and returns a list whose n-th element is the matrix of the differential map d_n of the complex $C_*^c(X)$, where the keyword parameters *targets* and *sources* are lists taken from a homologically admissible Morse matching on *finspace*.

`chcm-h-regular-dvf` *finspace targets sources*
: The keyword parameters *targets* and *sources* are lists representing the sets of targets and sources of a homologically admissible Morse matching on *finspace*, respectively. This function returns a Kenzo `CHAIN-COMPLEX` representing the complex $C_*^c(X)$, by using the matrices in the output list of `h-regular-dif-dvf` applied to *finspace*.

`h-regular-homology-dvf` *finspace dvfield dim*
: In case the parameter *dim* is given, this function returns the homology group of *finspace* in the given dimension *dim* by means of $C_*^c(X)$, which is obtained by using *dvfield*. Otherwise, all the possibly non-trivial homology groups of *finspace* are returned by using the chain complex $C_*^c(X)$.

`h-regular-homology-dvf-generators` *finspace dvfield dim*
: This function returns the generators of the homology group given by calling the function `(h-regular-homology-dvf` *finspace dvfield dim*`)`.

2.3.3.2 Didactical example

We consider again the space `finite-space` of Subsection 2.2.4.2, that is, the finite topological space in Figure 2.8. Now, we apply the new method for computing the homology groups by defining a discrete vector field by considering again two alternatives: computing its barycentric subdivision and applying the methods directly on `finite-space`.

Computing the barycentric subdivision

Now, computing a homologically admissible Morse matching on `sd-finite-space` and its chain complex `chcm-h-regular-dvf`, we have:

```
> (setf sd-dvfield (dvfield sd-finite-space))
> (setf chcm-sd-dvf (chcm-h-regular-dvf sd-finite-space
                                        sd-dvfield))
[K9 Chain-Complex]
> (length (basis chcm-sd-dvf 0))
1
```

```
> (length (basis chcm-sd-dvf 1))
1
> (length (basis chcm-sd-dvf 2))
3
```

Note that there are five critical points respect to the computed admissible vector field, which represents a considerable reduction of the number of generators of the chain complex `chcm-sd`. Regarding to homology, by using the method `h-regular-homology-dvf`, we obtain:

```
> (h-regular-homology-dvf sd-finite-space sd-dvfield 0)
Component Z
> (h-regular-homology-dvf sd-finite-space sd-dvfield 1)
Component Z/2Z
> (h-regular-homology-dvf sd-finite-space sd-dvfield 2)
Component Z
Component Z
```

Applying the methods directly

We can also apply the reduction explained in Section 2.3 to this space. At first, we compute a homologically admissible Morse matching on `finite-space`, which is represented in Figure 2.10:

```
> (setf dvfield (dvfield finite-space))
((1 5) (2 7) (4 8) (6 13) (11 16) (9 17) (10 18))
```

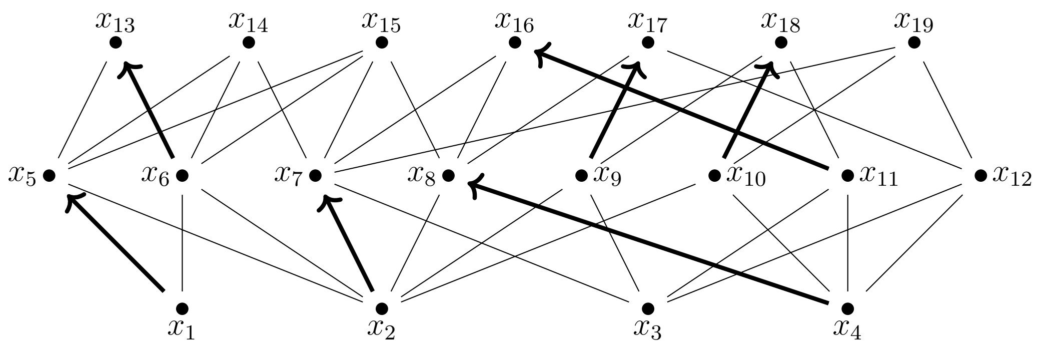


Figure 2.10: A homologically admissible Morse matching on `finite-space`.

Let us point out that the verification of the homological admissibility of the edges in the method `dvfield` that we have implemented (Subsection 2.3) is not trivial since, even when for finite spaces that are face posets of regular CW-complexes such verification is not necessary (because all the edges in that case are homologically admissible), for example in `finite-space` the subspace $\widehat{U}_{15} - \{x_8\}$ is not connected, then the edge (x_8, x_{15}) is not homologically admissible and therefore it can not be part of a Morse matching that we are searching.

The homology groups of `finite-space` can be computed by using the next commands:

```
> (h-regular-homology-dvf finite-space dvfield 0)
Component Z
> (h-regular-homology-dvf finite-space dvfield 1)
Component Z/2Z
> (h-regular-homology-dvf finite-space dvfield 2)
Component Z
Component Z
```

2.4 Performance analysis and comparison with related works

In order to compare the performance of our methods of Subsection 2.1.2 with those found in [Fer17a], we have generated 20 random spaces for each `n` (dimension) and `d` (density) appearing in Table 2.1, by using the function `posets.RandomPoset(n,d).relabel()` of SageMath [Dev20]; the average computing time (in seconds) of the core of each space in the respective sample of 20 spaces in Kenzo and in SageMath are the values shown in the table. We have made available our testing examples at [CR20] in the folder /h-regular-homology/data.

Table 2.2 follows the same procedure as described above but comparing the computing times given by the function `weak_core` in [Fer17a] with those given by our method `weakcore`. At this point, it is important to point out that, unlike the uniqueness (up to homeomorphism) of the core of a finite space, the compared functions in this table can provide, in general, different spaces with no weak points (even with different cardinalities as illustrated in Figure 2.4).

Regarding to compare the computation of homology of h-regular-spaces, we consider the barycentric subdivision X' of random spaces X as input of the function `homology` of [Fer17a] and of our method `h-regular-homology`. It is important to note that our algorithms allow to take advantage of the h-regular property of these spaces and therefore we do not need to compute their barycentric subdivisions again. For example, we have computed the homology of a space of 511 points obtained as the barycentric subdivision of a random poset of 9 points and density 0.8; such homology computations were performed in more than 17 hours by using the `homology` function, while our method did the same calculation in 15 seconds. In Table 2.3 we summarize some of the obtained results for small sizes and only for densities 0.2 and 0.4. In [Ren19], a method called `betti-of-poset` is defined; its implementation is similar to the function `homology` of [Fer17a] (both methods use the order complex of the space).

Finally, Table 2.4 shows the reductions on the size of different finite T_0-spaces when we compute homologically admissible Morse matchings on them, allowing to observe a better performance of our method `dvfield` in comparison with the outputs obtained by using the function `greedy_acyclic_matching` of [Fer17a].

Table 2.1: Average computing time (in seconds) of our method `core` in Kenzo (highlighted in bold) v.s. the function `core` in [Fer17a] on a sample of 20 random finite spaces, for each pair (`n`, `d`) with n = size and d = density.

d \ n	100	150	200	250	300	350	400	450	500
0.3	**0.051**	**0.095**	**0.129**	**0.202**	**0.208**	**0.420**	**0.461**	**0.858**	**0.858**
	0.381	0.838	1.444	2.292	3.285	4.408	5.801	7.290	9.781
0.4	**0.022**	**0.036**	**0.057**	**0.076**	**0.102**	**0.150**	**0.527**	**0.568**	**0.684**
	0.368	0.804	1.476	2.150	3.070	4.135	5.418	6.852	9.129
0.5	**0.018**	**0.034**	**0.054**	**0.075**	**0.112**	**0.153**	**0.464**	**0.609**	**0.807**
	0.352	0.767	1.339	2.038	2.924	3.946	5.136	6.538	8.128
0.6	**0.018**	**0.033**	**0.052**	**0.078**	**0.120**	**0.167**	**0.500**	**0.673**	**0.827**
	0.343	0.733	1.328	1.980	2.823	3.925	4.918	6.252	7.804
0.7	**0.019**	**0.032**	**0.055**	**0.082**	**0.123**	**0.178**	**0.560**	**0.724**	**0.998**
	0.327	0.712	1.215	1.884	2.707	3.739	4.743	6.174	7.460

Table 2.2: Average computing time (in seconds) of our method `weakcore` in Kenzo (highlighted in bold) v.s. the function `weak_core` in [Fer17a] on a sample of 20 random finite spaces, for each pair (`n`, `d`).

d \ n	10	20	30	40	50	60
0.2	**0.003**	**0.010**	**0.026**	**0.050**	**0.118**	**0.195**
	0.032	0.397	2.003	6.169	16.056	32.298
0.3	**0.003**	**0.008**	**0.024**	**0.042**	**0.037**	**0.027**
	0.053	0.553	1.929	6.650	14.555	27.732
0.4	**0.002**	**0.005**	**0.010**	**0.009**	**0.013**	**0.013**
	0.054	0.564	2.010	5.083	12.471	20.516
0.5	**0.002**	**0.004**	**0.007**	**0.006**	**0.008**	**0.009**
	0.061	0.446	1.606	4.198	8.640	15.933
0.6	**0.001**	**0.003**	**0.004**	**0.006**	**0.007**	**0.010**
	0.051	0.353	1.178	3.169	6.041	11.393
0.7	**0.001**	**0.003**	**0.004**	**0.005**	**0.006**	**0.009**
	0.045	0.312	0.906	2.275	4.463	7.609
0.8	**0.001**	**0.002**	**0.004**	**0.005**	**0.006**	**0.009**
	0.040	0.254	0.784	1.577	2.856	5.105

Table 2.3: Average computing time (in seconds) of our method `h-regular-homology` in Kenzo (highlighted in bold) vs the function `homology` in [Fer17a] on a sample of barycentric subdivisions of 20 random finite spaces, for each pair (`n`, `d`).

d \ n	9	10	11	12	13	14
0.2	**0.032**	**0.069**	**0.156**	**0.155**	**0.318**	**1.230**
	0.025	0.066	0.946	0.220	4.836	197.137
0.4	**0.228**	**0.740**	**2.380**	**3.403**	**8.314**	**15.017**
	0.491	14.247	127.414	184.221	1380.388	2774.636

Table 2.4: Comparison of the average number of critical points and the average computation time (in seconds) of a homologically admissible Morse matching given by our method `dvfield` (highlighted in bold) v.s. the one given by the function `greedy_acyclic_matching` [Fer17a], applied to the barycentric subdivisions of a sample of 20 random finite topological spaces for each pair (`n`, `d`).

Size `n`	*Density* `d`	*Size* X'	*Critical points*		*Computing time*	
9	0.2	26.8	5.6	**2.8**	0.003	**0.003**
	0.4	57.0	8.2	**2.0**	0.011	**0.013**
	0.6	176.9	21.0	**1.1**	0.111	**0.087**
	0.8	349.6	44.7	**1.0**	0.437	**0.275**
10	0.2	36.4	7.4	**3.0**	0.004	**0.005**
	0.4	101.6	13.4	**1.7**	0.036	**0.035**
	0.6	264.2	31.8	**1.1**	0.260	**0.191**
	0.8	627.2	80.6	**1.0**	1.658	**0.870**
11	0.2	46.8	7.7	**3.0**	0.008	**0.010**
	0.4	189.4	23.1	**1.5**	0.123	**0.107**
	0.6	434.9	54.0	**1.3**	0.727	**0.469**
	0.8	1057.0	141.2	**1.0**	5.326	**2.420**
12	0.2	55.1	9.6	**3.4**	0.009	**0.011**
	0.4	229.2	29.1	**2.5**	0.182	**0.157**
	0.6	829.7	107.3	**1.0**	3.857	**1.860**
	0.8	2039.8	292.6	**1.0**	21.831	**8.464**
13	0.2	75.4	11.9	**3.0**	0.015	**0.019**
	0.4	358.6	46.8	**1.6**	0.462	**0.346**
	0.6	1432.2	189.6	**1.1**	11.184	**4.857**
	0.8	3359.0	503.9	**1.0**	78.325	**23.415**
14	0.2	110.1	15.4	**2.8**	0.067	**0.053**
	0.4	501.6	63.0	**1.7**	1.057	**0.676**
	0.6	2426.0	355.1	**1.0**	47.102	**15.645**
	0.8	7988.4	1394.6	**1.0**	864.377	**156.399**

Chapter 3

h-regularization of finite topological spaces

In Chapter 2 we have shown some algorithms to compute homotopical invariants of finite topological spaces and we have developed constructive versions of theoretical results found in [CO17],[Min12] in order to compute homology groups in the class of h-regular spaces. Until now in the literature, examples of h-regular spaces are restricted to face posets of h-regular complexes and regular CW-complexes (in particular, simplicial complexes). Indeed, homology groups of a finite T_0-space X are computed by using algorithms designed to be applied on simplicial complexes as in [Fer17b], [Ren19], so that the order complex $\mathcal{K}(X)$ is used to find such invariants of X (by Theorem 1.39). Therefore, the barycentric subdivision $X' = \mathcal{X}(\mathcal{K}(X))$ is an h-regular space which, in principle, we can take as input of our algorithms. However, barycentric subdivisions can get quite large, since each n-maximal chain of the original finite T_0-space produces $(n+1)!$ chains in the subdivision. In this way, it is natural to ask us whether there exist h-regular spaces, other than X', weak homotopy equivalent to X but with a smaller number of elements than X' that we can use as input of our implemented algorithms.

In this chapter, we will show a procedure to *h-regularize* a finite T_0-space X of height at most 2, i.e. we will describe an algorithmic process to find an h-regular space X_h which is simple homotopy equivalent to X. The key point of this process will consist of separating wedges of spheres which come from minimal open sets of the elements having height 2. In this way, the space X_h will contain more points and edges than X, but it will be smaller than X'. Moreover, we give examples of h-regularizations of finite T_0-spaces by using our implementation of the method in the Kenzo system.

The work explained in this chapter has been presented in the conference [CR19b] and the preprint [CRLRS20b]. The code of the developed Lisp functions is included at [CR20] in the folder /h-regularization.

3.1 Some properties of h-regular spaces

We need to recall the definition of h-regular spaces (Definition 1.48): a finite T_0-space X is ***h-regular*** if $\widehat{U}_x \overset{\text{we}}{\approx} S^{h(x)}$, for every $x \in X$. In this section we state some results about h-regular spaces. In particular, h-regular spaces of height 1 are characterized, which is a useful tool for developing the *h-regularization* process described in Section 3.3.

Definition 3.1. Let X be a finite T_0-space. We say that X is ***n-h-regular*** if for $n \in \mathbb{N}$, the subposet $X^{(n)} = \{x \in X : h(x) \leqslant n\}$ is h-regular.

The next lemma is a simple consequence of the definitions just mentioned.

Lemma 3.2. *Let X be a n-h-regular space for some $n \in \mathbb{N}_0$ and let $A \subseteq X$ be an open subset. Then A is n-h-regular. In particular, for all $x \in X$, U_x^X and $\widehat{U}_x^X$ are n-h-regular spaces.*

Proof. Let $a \in A^{(n)}$. By hypothesis, $\widehat{U}_a^X \overset{\text{we}}{\approx} S^{h^X(a)-1}$. Observe that, since A is a down-set (if $b \in A$ and $c \leqslant b$ then $c \in A$), $h^A(a) = h^X(a)$, $U_a^A = U_a^X$ and therefore $A^{(n)}$ is a subspace of $X^{(n)}$. Then $\widehat{U}_a^A \overset{\text{we}}{\approx} S^{h^A(x)-1}$, hence A is n-h-regular. ■

The core of an h-regular space is not h-regular in general. Nevertheless, in spaces of height 1, the h-regularity property is inherited by its cores.

Lemma 3.3. *Let X be an h-regular space of height 1 and let X_c be a core of X. Then X_c is h-regular.*

Proof. Without loss of generality, assume that X is connected (otherwise consider each connected component independently). If X is contractible there is nothing to proof. Suppose that X is not contractible and let $x \in \text{mxl}(X_c) \subseteq \text{mxl}(X)$. Since X_c is a subspace of X, $\widehat{U}_x^{X_c} = \widehat{U}_x^X \cap X_c$ so that $\#\widehat{U}_x^{X_c} \leqslant \#\widehat{U}_x^X = 2$. Since $h(x) = 1$ then $\#\widehat{U}_x^{X_c} \geqslant 1$, but X_c has no beat points so that necessarily $\#\widehat{U}_x^{X_c} = 2$. ■

If X is a connected finite T_0-space of height 1, $|\mathcal{K}(X)|$ is a connected graph so that $X \overset{\text{we}}{\approx} \bigvee_{i=1}^{q} S^1$ if and only if $\mathcal{X}(X) = 1 - q$, where $\mathcal{X}(X) = \#X - \#\text{E}(\mathcal{H}(X))$ is the Euler characteristic of X.

Observe the Euler characteristic characterizes the weak homotopy type of a connected graph. It is possible to characterize the h-regular spaces of height 1 by using this well-known fact.

Lemma 3.4. *Let X be an h-regular connected finite T_0-space of height 1. Then*

$$X \overset{we}{\approx} \bigvee_{i=1}^{q} S^1 \text{ if and only if } \#\text{mnl}(X) - \#\text{mxl}(X) = 1 - q.$$

Proof. Since X has height 1, $\#X = \#\text{mxl}(X) + \#\text{mnl}(X)$ and by the h-regular property, for each $z \in \text{mxl}(X)$ there are exactly two edges with z as head, hence $\#\text{E}(\mathcal{H}(X)) = 2\#\text{mxl}(X)$ and therefore $\mathcal{X}(X) = \#\text{mnl}(X) - \#\text{mxl}(X)$. ■

Lemma 3.5. *Let X be an h-regular connected finite T_0-space of height 1. If X^{op} is h-regular then $X \overset{we}{\approx} S^1$. Moreover, the converse is true if X has no beat points, and in this case $X^{op} \overset{hom}{\approx} X$.*

Proof. If X^{op} is h-regular, then $\#\text{E}(\mathcal{H}(X^{op})) = 2\#\text{mxl}(X^{op})$ and by the h-regularity of X we have

$$2\#\text{mnl}(X) = 2\#\text{mxl}(X^{op}) = \#\text{E}(\mathcal{H}(X^{op})) = \#\text{E}(\mathcal{H}(X)) = 2\#\text{mxl}(X),$$

hence $X \overset{\text{we}}{\approx} S^1$ by Lemma 3.4.

Now, suppose that X has no beat points and $X \overset{\text{we}}{\approx} S^1$. Consider the following partition of $\text{mnl}(X)$:

$$A = \{x \in \text{mnl}(X) : \#\widehat{F}_x^X = 2\}\ ,\ B = \{x \in \text{mnl}(X) : \#\widehat{F}_x^X > 2\}.$$

Since X is h-regular and using Lemma 3.4, $\#\text{E}(\mathcal{H}(X)) = 2\#\text{mxl}(X) = 2\#\text{mnl}(X)$. Note that if $B \neq \emptyset$ then

$$\#\text{E}(\mathcal{H}(X)) = \sum_{x\in\text{mnl}(X)} \#\widehat{F}_x^X = \sum_{x\in A} \#\widehat{F}_x^X + \sum_{x\in B} \#\widehat{F}_x^X > 2\#A + 2\#B = 2\#\text{mnl}(X),$$

which is absurd, hence $B = \emptyset$ and therefore $\text{mnl}(X) = A$. Since $\text{mxl}(X^{op}) = \text{mnl}(X)$, every $x \in \text{mxl}(X^{op})$ satisfies $\#\widehat{F}_x^X = \#\widehat{U}_x^{X^{op}} = 2$ which is equivalent to say that X^{op} is h-regular. ■

Lemma 3.5 allows us to establish the following definition.

Definition 3.6. A ***2m-crown*** is an h-regular finite model of S^1 of height 1 without beat points whose cardinal is $2m$. The Hasse diagram of a $2m$-crown looks like in Figure 3.1.

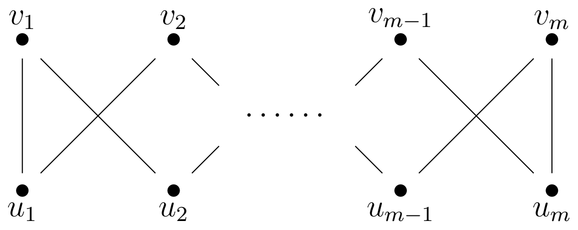


Figure 3.1: Hasse diagram of a $2m$-crown.

Crown spaces are going to play a crucial role in the h-regularization process. Observe that if a point is deleted from a crown (which is a connected space), the resulting space continues being connected. More generally, when a point of a crown that is contained in an h-regular connected finite T_0-space Z is deleted from Z, the connectedness is maintained.

Lemma 3.7. *Let Z be an h-regular connected finite T_0-space of height 1 and let Y be a $2m$-crown contained in Z. If $x \in \operatorname{mxl}(Y)$ then $Z \smallsetminus \{x\}$ is connected.*

Proof. Since Y is a $2m$-crown, without loss of generality we assume that $x = v_1$ (see Figure 3.1). It is sufficient to show that for any $z \in Z \smallsetminus \{v_1\}$ there exists a fence of elements in $Z \smallsetminus \{v_1\}$ connecting z with u_1. Since Z is connected, there exist $x_1, \dots, x_t \in Z$ such that

$$z = x_1 < x_2 > \cdots < x_{t-1} > x_t = u_1. \tag{3.1}$$

Consider the set $L = \{k : x_k = v_1\}$. If $L = \emptyset$, then (3.1) is a fence of elements in $Z \smallsetminus \{v_1\}$ connecting z and u_1; otherwise take $l = \min L$. Observe that by the h-regularity of Z, necessarily $\{x_{l-1}, x_{l+1}\} = \{u_1, u_2\}$. Then, by the minimality of l:

- If $x_{l-1} = u_1$, the fence $z = x_1 < x_2 > \cdots < x_{l-1} = u_1$ connects z and u_1 in $Z \smallsetminus \{v_1\}$.
- If $x_{l-1} = u_2$, the fence

$$\begin{aligned} z &= x_1 < x_2 > \cdots < x_{l-1} \\ &= u_2 < v_3 > u_4 < \cdots < v_m > u_m < v_{m-1} > \cdots < v_2 > u_1 \end{aligned}$$

  connects z and u_1 in $Z \smallsetminus \{v_1\}$.

Therefore $Z \smallsetminus \{x\}$ is connected. ■

Corollary 3.8. *Let X be a $2m$-crown and $x \in X$. Then $X \smallsetminus \{x\}$ is connected.*

Proof. If $x \in \operatorname{mxl}(X)$, the result follows from Lemma 3.7 by taking $Z = X$ and $Y = X$. If $x \in \operatorname{mnl}(X)$, applying Lemma 3.7 for $Z = X^{op}$ and $Y = X^{op}$ (this choice is valid because X^{op} is h-regular by Lemma 3.5), we obtain that $X^{op} \smallsetminus \{x\}$ is connected and therefore $X \smallsetminus \{x\}$ is also connected. ■

3.2 Glueable pairs

Some of the reductions that can be found in the literature, in order to obtain weak homotopy equivalent spaces to a given finite T_0-space, are contained in the next three definitions.

Definition 3.9. (Section 11.2, [Bar11]) Let X be a finite T_0-space of height at most 2 and let $a, b \in X$ be two maximal elements of X such that $U_a \cap U_b$ is contractible. We say that there is a ***qc-reduction*** from X to $Y \smallsetminus \{a, b\}$ where $Y = X \cup \{c\}$ with $a < c > b$. We say that X is ***qc-reducible*** if we can obtain a space with a maximum by performing qc-reductions starting from X.

Definition 3.10. (Definition 3.2.5, [Fer17b]) Let X be a finite T_0-space of height at most 2 and let $a, b \in X$ be neither maximal nor minimal points such that $U_a \cap U_b = \{*\}$. If for every $x \in F_a \smallsetminus F_b$, $U_b \cap U_x = \{*\}$, and for every $x \in F_b \smallsetminus F_a$, $U_a \cap U_x = \{*\}$, we say that there is a ***middle-reduction*** from X to the quotient $X/\{a, b\}$. We say that X is ***middle-reducible*** if it can be transformed into a connected space with a unique point (of height 1) by performing middle-reductions.

Definition 3.11. (Definition 3.2.10, [Fer17b]) Let X be a finite T_0-space and let $e = (a, b)$ be an edge in the Hasse diagram $\mathcal{H}(X)$, with b a maximal element. If $U_b \smallsetminus e$ is contractible, we say that there is an ***edge-reduction*** from X to $X \smallsetminus e$.

The above definitions are inspired in Theorem 1.39, a standard tool used to guarantee the existence of a weak homotopy equivalence between two finite T_0-spaces. We have adopted the term *glueable pair* to refer to a more general kind of reduction.

Definition 3.12. Let X be a finite T_0-space. A subset $A = \{a, b\} \subset X$ is a ***glueable pair*** if it satisfies the following conditions:

1. $\widehat{U}_a \cap \widehat{F}_b = \emptyset$ and $\widehat{U}_b \cap \widehat{F}_a = \emptyset$.
2. $U_x \cup U_b$ is homotopically trivial for every $x \in F_a \smallsetminus F_b$ and $U_a \cup U_x$ is homotopically trivial for every $x \in F_b \smallsetminus F_a$.

Note that if $\{a, b\}$ is a glueable pair, $U_a \cup U_b$ is homotopically trivial. In the particular case when $U_a \cup U_b$ is contractible, $U_a \cap U_b$ is also contractible as stated in the next Proposition.

Proposition 3.13. [CO18] *Let X be a finite T_0-space and let $a, b \in X$. Then $U_a \cup U_b$ is homotopy equivalent to the non-Hausdorff suspension of $U_a \cap U_b$. In particular, if $U_a \cap U_b$ is contractible then $U_a \cup U_b$ is contractible.*

The following result shows that the quotient of a space by a glueable pair does not modify its weak homotopy type.

Proposition 3.14. *Let X be a finite T_0-space and $A = \{a, b\} \subset X$ a glueable pair. Then $q : X \longrightarrow X/A$ is a weak homotopy equivalence.*

Proof. Observe that $\underline{A} = U_a \cup U_b$ and $\overline{A} = F_a \cup F_b$, then by condition 1 in Definition 3.12 it is clear that $\underline{A} \cap \overline{A} = A$. Hence X/A is a finite T_0-space by Lemma 1.13. In order to prove that $q : X \longrightarrow X/A$ is a weak homotopy equivalence, it is sufficient to show that for each $x \in X$, the restricted map $q|_{q^{-1}(U_{q(x)})} : q^{-1}(U_{q(x)}) \longrightarrow U_{q(x)}$ is a weak homotopy equivalence (Theorem 1.39). For this purpose we have two cases:

1. If $x \notin \overline{A}$, by Lemma 1.14, $U_{q(x)} = q(U_x)$. Since $x \notin F_a \cup F_b$ then $a \notin U_x$ and $b \notin U_x$, hence $q^{-1}(q(U_x)) = U_x$.

2. If $x \in \overline{A}$, by Lemma 1.14, $U_{q(x)} = q(U_x \cup \underline{A})$. Note that if $x \in F_a \cap F_b$, $U_a \cup U_b \subseteq U_x$ then $q^{-1}(U_{q(x)}) = U_x$. Observe that $q^{-1}(q(U_x \cup U_b)) = U_x \cup U_b$ if $x \in F_a \smallsetminus F_b$ and $q^{-1}(q(U_a \cup U_x)) = U_a \cup U_x$ if $x \in F_b \smallsetminus F_a$. Therefore, for every $x \in \overline{A}$, $q^{-1}(U_{q(x)})$ is homotopically trivial by condition 2 in Definition 3.12.

In any case, the map $q|_{q^{-1}(U_{q(x)})} : q^{-1}(U_{q(x)}) \longrightarrow U_{q(x)}$ is a weak homotopy equivalence for each $x \in X$ as we wanted. ■

Example 3.15. Figure 3.2 shows how to obtain a weak homotopy equivalent space to X (the face poset of the boundary of a triangle seen as a simplicial complex) by taking the quotient by the glueable pair $A = \{a, b\}$; after that, we delete A as a beat point of X/A.

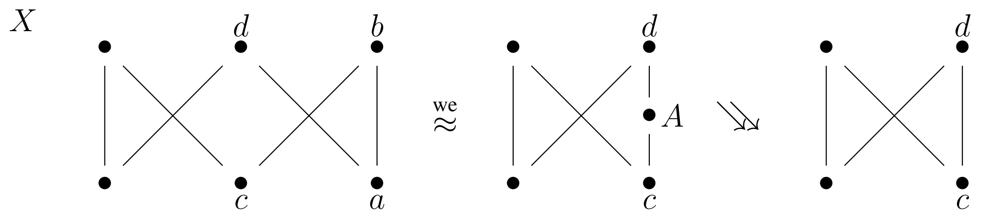


Figure 3.2: Quotient of X by the glueable pair $A = \{a, b\}$ followed by an elementary strong collapse.

Corollary 3.16. *If X is a finite T_0-space and $a, b \in X$ satisfy that $\widehat{F}_a = \widehat{F}_b$ and $U_a \cup U_b$ is homotopically trivial, then $X \overset{we}{\approx} X/\{a, b\}$.*

Above Corollary is a generalization of the next Proposition.

Proposition 3.17. [CO18] *Let X be a finite T_0-space and let a and b be maximal elements of X. If $U_a \cup U_b$ is homotopically trivial then the quotient map $q : X \longrightarrow X/\{a, b\}$ is a weak homotopy equivalence.*

Some remarks can be derived from Proposition 3.14:

- Suppose that b is a down beat point of X and $a = \max(\widehat{U}_b)$. In particular, (a, b) is an edge of the Hasse diagram of X hence $\widehat{U}_a \cap \widehat{F}_b = \emptyset$ and $\widehat{U}_b \cap \widehat{F}_a = \emptyset$. In addition, $U_a \cup U_b = U_b$ is contractible and therefore $\{a, b\}$ is a glueable pair. A similar argument works when b is an up beat point of X and $a = \min(\widehat{F}_b)$. It means that elementary strong collapses in finite T_0-spaces can be seen as quotients by glueable pairs.

- If $a, b \in X$ are maximal elements, $\widehat{F}_a = \emptyset = \widehat{F}_b$, then Proposition 3.14 generalizes Proposition 3.17. Bearing in mind this fact, edge-reductions can be seen as a sequence of quotients by some glueable pairs: if b is a maximal element of X and $e = (c, b)$ is an edge of $\mathcal{H}(X)$ such that $U_b^X \smallsetminus e$ is contractible (as in Definition 3.11), we consider the space $Y = (X \smallsetminus e) \cup \{a\}$ where (c, a) is an edge of $\mathcal{H}(Y)$ and a is a down beat point

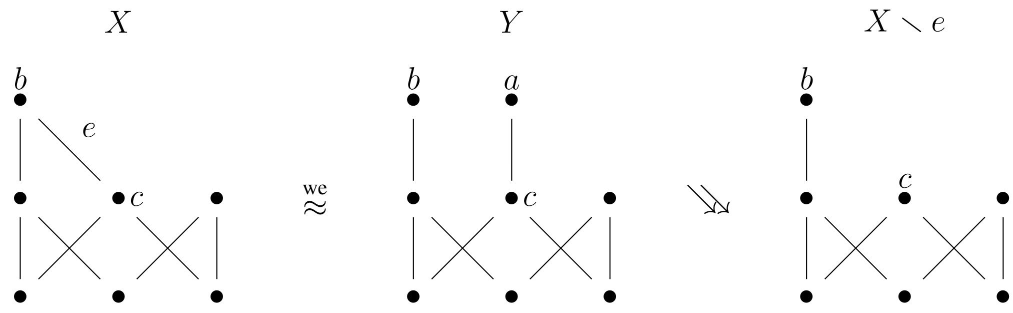


Figure 3.3: An edge-reduction seen as a quotient by a glueable pair followed by an elementary strong collapse.

of Y, then the maximal elements $\{a,b\}$ form a glueable pair of Y since $U_a^Y \cup U_b^Y = \{a\} \cup (U_b^X \smallsetminus e) \searrow\!\!\searrow U_b^X \smallsetminus e \searrow\!\!\searrow *$ and therefore $X \smallsetminus e \nearrow\!\!\nearrow Y \overset{\text{we}}{\approx} Y/\{a,b\} \overset{\text{hom}}{\approx} X$. The sequence of spaces in Figure 3.3 is an example of this situation.

- Regarding finite spaces of height at most 2, qc-reductions can be seen as quotients by glueable pairs (by definition) and the same occurs for middle-reductions: if $a, b \in X$ are neither maximal nor minimal points and $U_a \cap U_b = \{*\}$ (as in Definition 3.10), then can not exist $x \in X$ such that $a < x < b$ or $b < x < a$ (in other case a or b is either maximal or minimal).

Now, we are going to describe the opposite process: instead of gluing a pair (which decreases in one point the given space), we add a point (under some conditions). The following lemma captures this idea and it will be useful in the next sections, where this *inverse method* (to that of reduction by taking a quotient by a glueable pair) will be used to modify a finite space until an h-regular space, which is simple homotopy equivalent to the given one, is obtained.

Lemma 3.18. *Let X be a finite T_0-space and $a \in X$. Consider the set $Z = X \cup \{b\}$, where $b \notin X$, and suppose that Z is endowed with a topology satisfying the following properties:*

1. $\widehat{F}_a^X = \widehat{F}_a^Z = \widehat{F}_b^Z$,
2. $\widehat{U}_a^X = \widehat{U}_a^Z \cup \widehat{U}_b^Z$,
3. *For* $x \in X \smallsetminus \{a\}$, $U_x^Z = \begin{cases} U_x^X \cup \{b\} & , x \in \widehat{F}_a^X \\ U_x^X & , x \notin \widehat{F}_a^X \end{cases}$
4. $U_a^Z \cup U_b^Z$ *is homotopically trivial.*

Then Z is weak homotopy equivalent to X. Moreover, if $U_a^Z \cup U_b^Z$ is contractible then $Z \diagup\!\!\searrow X$.

Proof. Properties *1* and *4* say that the map $q : Z \longrightarrow Z/\{a,b\}$ is a weak homotopy equivalence (by Corollary 3.16). It is direct to show that, if $\iota : X \hookrightarrow Z$ is the inclusion map, the map $q\iota : X \longrightarrow Z/\{a,b\}$ is a homeomorphism, which is sufficient to obtain $Z \overset{\text{we}}{\approx} X$.

Now, if $U_a^Z \cup U_b^Z$ is contractible, consider a space $M = Z \cup \{c\}$ where $F_a^M = F_b^M = \widehat{F}_a^Z \cup \{c\}$ and $\widehat{U}_c^M = U_a^Z \cup U_b^Z$ (see Figure 3.4).

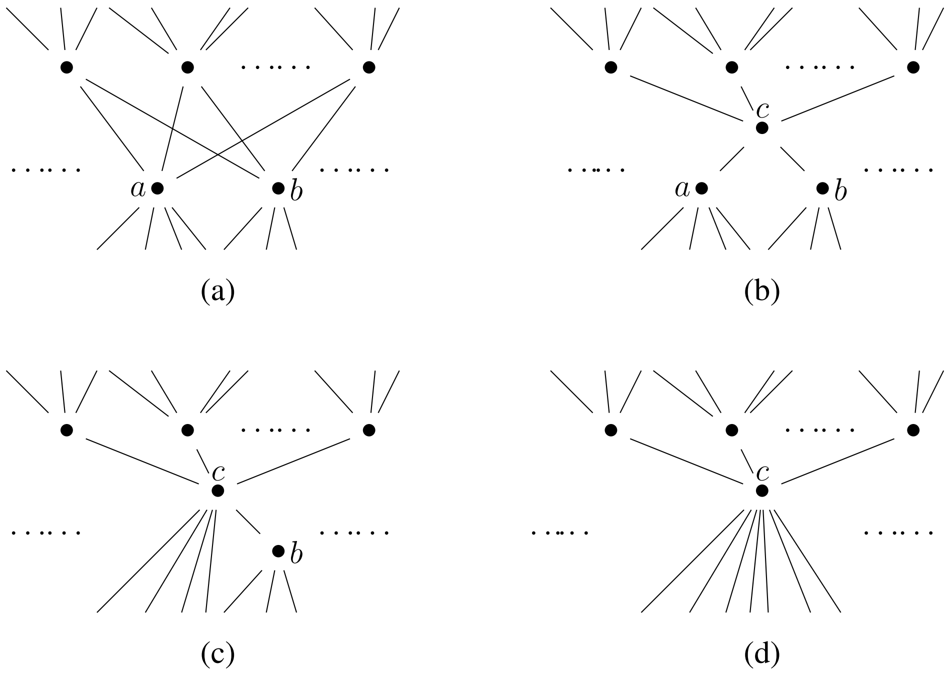


Figure 3.4: Illustration of: (a) $C_a^Z \cup C_b^Z$, (b) C_c^M, (c) $C_c^{M \smallsetminus \{a\}}$, (d) $C_c^{M \smallsetminus \{a,b\}}$.

Note that c is a weak point of M and a, b are beat points of M. Then $Z \nearrow M \searrow\!\!\searrow M \smallsetminus \{a\} \searrow\!\!\searrow M \smallsetminus \{a,b\}$. Finally, observe that $M \smallsetminus \{a,b\} \overset{\text{hom}}{\approx} X$ (the homeomorphism replaces the label c in $M \smallsetminus \{a,b\}$ by the label a), then $Z \diagup\!\!\searrow X$. ■

Remark 3.19. Thanks to property *3* in the statement of Lemma 3.18, it is easy to verify that closures and minimal open sets coincide in X and Z, that is, for each $x \in X$, $U_x^Z = U_x^X$ and $F_x^Z = F_x^X$.

3.3 h-regularization of finite spaces

In this section, we show an effective method to construct an h-regular space simple homotopy equivalent to a given finite T_0-space of height at most 2. Along this section, we are going to illustrate the application of our results on the finite T_0-space M in Figure 3.5 (we will *h-regularize* the space M). Note that M is not h-regular since, for example, $h(i) = 1$ but $\#\widehat{U}_i = 3$.

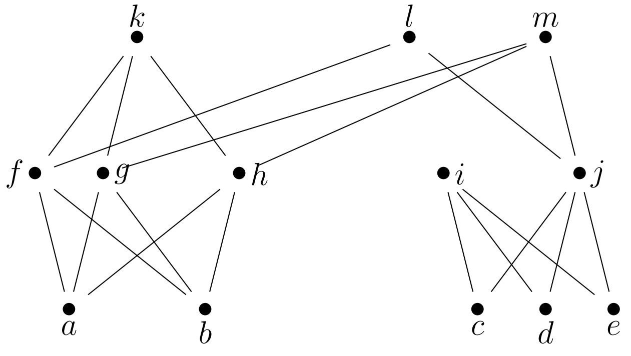


Figure 3.5: Hasse diagram of the finite T_0-space M.

The process starts modifying the minimal open sets of the elements of height 1. This procedure does not affect the elements of higher heights, so we adopt the following notation to emphasize this:

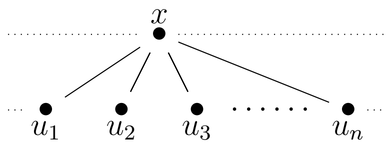
Notation 3.20. If X and Y are finite T_0-spaces such that $X \smallsetminus X^{(n)} \overset{\text{hom}}{\approx} Y \smallsetminus Y^{(n)}$ and $X \nearrow\!\searrow Y$ for some $n \in \mathbb{N}$, we will write $X \approx_n Y$.

As we have previously established, each element x of height 1 in an h-regular space must satisfy the condition $\#\widehat{U}_x = 2$ (the 0-dimensional sphere is a discrete two-element set). The next lemma allows us to separate the elements of $\widehat{U}_x$ when $\#\widehat{U}_x > 2$.

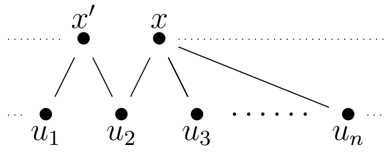
Lemma 3.21. *Let X be a finite T_0-space and let $x \in X$ such that $h(x) = 1$ and $\#\widehat{U}^X_x = n \geqslant 3$. Then there exists a finite T_0-space X_x such that $\#\widehat{U}^{X_x}_x = n - 1$ and $X \approx_1 X_x$.*

Proof. Let $\widehat{U}^X_x = \{u_1, \dots, u_n\} \subseteq \operatorname{mnl}(X)$. Consider the space $X_x = X \cup \{x'\}$ whose minimal basis $\{U^{X_x}_z\}_{z \in X_x}$ is given by:

$X^{(1)} =$



$X^{(1)}_x =$



$$, \quad \widehat{U}^{X_x}_z = \begin{cases} \{u_1, u_2\} & , \ z = x' \\ \{u_2, \dots, u_n\} & , \ z = x \\ \widehat{U}^X_z \cup \{x'\} & , \ z \in \widehat{F}^X_x \\ \widehat{U}^X_z & , \ z \notin \widehat{F}^X_x . \end{cases}$$

By construction, $\{x, x'\}$ is an antichain of X_x such that

$$z \in \widehat{F}^{X_x}_x \iff x \in \widehat{U}^{X_x}_z \iff z \in \widehat{F}^X_x,$$

$$z \in \widehat{F}^{X_x}_{x'} \iff x' \in \widehat{U}^{X_x}_z \iff z \in \widehat{F}^X_x,$$

therefore $\widehat{F}_x^X = \widehat{F}_x^{X_x} = \widehat{F}_{x'}^{X_x}$. Also, by Proposition 3.13, $U_x^{X_x} \cup U_{x'}^{X_x}$ is contractible because $U_x^{X_x} \cap U_{x'}^{X_x} = \{u_2\}$ is contractible, hence by Lemma 3.18 we have $X_x \diagup\!\!\searrow X$. Finally, observe that $Z := X - X^{(1)}$ equals $X_x - X_x^{(1)}$ as sets; moreover, for all $y \in Z$, since $x' \notin Z$ then $U_y^Z = U_y^X \cap Z = U_y^{X_x} \cap Z$. Then, the topologies of X and X_x coincide on Z. ■

Corollary 3.22. *Let X be a finite T_0-space and let $x \in X$ such that $h(x) = 1$ and $\#\widehat{U}_x^X = n \geqslant 3$. Then there exists a finite T_0-space X_x such that $\#\widehat{U}_x^{X_x} = 2$ and $X \approx_1 X_x$.*

Proof. By applying Lemma 3.21 succesive times, we can construct a sequence of spaces

$$X = X_{x,n} \subset X_{x,n-1} \subset \cdots \subset X_{x,2}$$

such that for each $k = 3, \ldots, n$ we have $\#\widehat{U}_x^{X_{x,k}} = k$ and $X_{x,k} \approx_1 X_{x,k-1}$. Taking $X_x = X_{x,2}$ the result is obtained. ■

The above result permits to modify the tilded minimal open set of a point of height 1 in X not satisfying the h-regular property. Applying this process to each point of height 1, we can find a 1-h-regular space simple homotopy equivalent to X.

Proposition 3.23. *Let X be a connected finite T_0-space of height greater than zero, without beat points. Then there exists a 1-h-regular connected finite T_0-space L without beat points such that $X \approx_1 L$.*

Proof. If X is 1-h-regular, take $L = X$; otherwise there exist $v_1, \ldots, v_r \in X$ elements of height 1 such that $\#\widehat{U}_{v_k}^X = n_k \geqslant 3$ for each $k = 1, \ldots, r$ (observe that we have used the non existence of beat points in X). Applying Corollary 3.22 to X and v_1, we obtain a space X_1 such that $\#\widehat{U}_{v_1}^{X_1} = 2$ and $X \approx_1 X_1$. Succesive applications of that corollary allow us to construct a sequence of spaces

$$X = X_0 \subset X_1 \subset \cdots \subset X_r$$

such that for each $k = 1, \ldots, r$ we have $\#\widehat{U}_{v_i}^{X_k} = 2$ for all $i = 1, \ldots, k$, X_k has no beat points (by Remark 3.19) and $X_{k-1} \approx_1 X_k$. Taking $L = X_r$ the result is obtained. ■

Theorem 3.24. *Let X be a finite T_0-space without beat points. There exists a 1-h-regular space L without beat points such that $X \approx_1 L$.*

Proof. If X is 1-h-regular, take $L = X$; otherwise consider $X = \bigsqcup_{k=1}^m C_k$ where the C_k are the connected components of X. If $h(C_k) = 0$, C_k is a single point and we take $L_k = C_k$; if $h(C_k) \geqslant 1$, by Proposition 3.23 there exists a 1-h-regular connected finite T_0-space without beat points L_k such that $C_k \approx_1 L_k$. Defining $L = \bigsqcup_{k=1}^m L_k$ we obtain:

$$X = \bigsqcup_{k=1}^{m} C_k \diagup\!\!\searrow \bigsqcup_{k=1}^{m} L_k = L$$

$$X - X^{(1)} = \bigsqcup_{k=1}^{m} \left(C_k - C_k^{(1)}\right) \overset{\text{hom}}{\approx} \bigsqcup_{k=1}^{m} \left(L_k - L_k^{(1)}\right) = L - L^{(1)}.$$

■

Regarding the space M in Figure 3.5, observe that $\#\widehat{U}_i = 3$ and $\#\widehat{U}_j = 3$, therefore we apply Theorem 3.24 obtaining the 1-h-regular space in Figure 3.6 (we have *1-h-regularized* the space M).

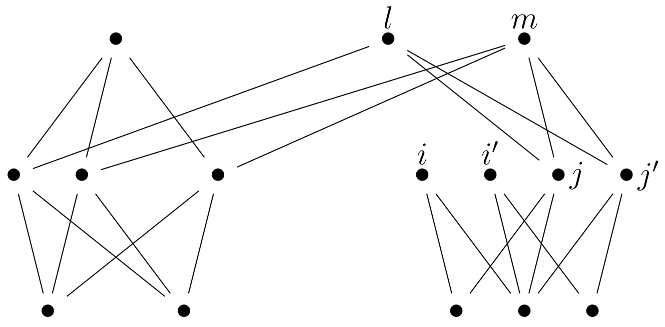


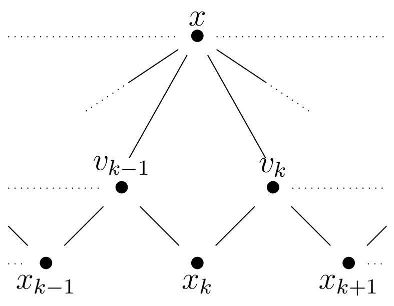
Figure 3.6: 1-h-regularization of M.

Once we have modified a given space so that its elements of height 1 satisfy the h-regularity property, we have to achieve that for each element x of height 2, the subspace $\widehat{U}_x$ to be weak homotopy equivalent to S^1. In particular, $\widehat{U}_x$ must be connected; if it is not the case, the following result allows us to modify the space in order to solve this local situation.

Proposition 3.25. *Let X be a finite T_0-space and let $x \in X$ with $h(x) \geqslant 2$. Then there exists a finite T_0-space X_1 such that $\widehat{U}_x^{X_1}$ is connected and $X \approx_1 X_1$. Moreover for $n \in \mathbb{N}_0$, if X is n-h-regular then so is X_1.*

Proof. If $\widehat{U}_x^X$ is connected, take $X_1 = X$; otherwise consider $\widehat{U}_x^X = \bigsqcup_{k=1}^m C_k$ where C_k are the connected components of $\widehat{U}_x^X$. Since $C_k \neq \emptyset$ fix any $x_k \in \text{mnl}(C_k)$ for each $k = 1, \dots, m$. Consider the T_0-space $X_1 = X \cup \{v_1, \dots, v_{m-1}\}$ whose minimal basis $\{U_z^{X_1}\}_{z \in X_1}$ is defined by:

$$\widehat{U}_z^{X_1} = \begin{cases} \{x_k, x_{k+1}\} & , \ z = v_k \text{ for } k = 1, \dots, m-1 \\ \widehat{U}_z^X \cup \{v_1, \dots, v_{m-1}\} & , \ z \in F_x^X \\ \widehat{U}_z^X & , \ z \notin F_x^X. \end{cases}$$

$X_1^{(h(x))} =$


Observe that for each $k = 1, \dots, m-1$, $\widehat{F}_{v_k}^{X_1} = F_x^X$ ($z \in \widehat{F}_{v_k}^{X_1} \iff v_k \in \widehat{U}_z^{X_1} \iff z \in F_x^X$). This means that v_k is an up beat point of X_1 and then $X_1 \searrow\!\!\searrow X$. To prove the connectedness of $\widehat{U}_x^{X_1}$ it is sufficient to observe that $x_1 < v_1 > x_2 < \cdots > x_{m-1} < v_{m-1} > x_m$ is a fence in X_1 connecting all the C_k. Finally, by construction, $X_1^{(0)} = X^{(0)}$, $X - X^{(1)} \overset{\text{hom}}{\approx} X_1 - X_1^{(1)}$ and for $n \geqslant 1$, $X_1^{(n)} = X^{(n)} \cup \{v_1, \dots, v_{m-1}\}$, $h(v_k) = 1$ and $\#\widehat{U}_{v_k}^{X_1} = 2$, therefore if X is n-h-regular, so is X_1. ■

Bearing in mind the 1-h-regular space in Figure 3.6, we can use Proposition 3.25 in order to make $\widehat{U}_l$ and $\widehat{U}_m$ connected subspaces by adding the points l' and m' as it is shown in Figure 3.7, so we have obtained a 1-h-regular space with connected tilded minimal open sets for all the elements of height 2.

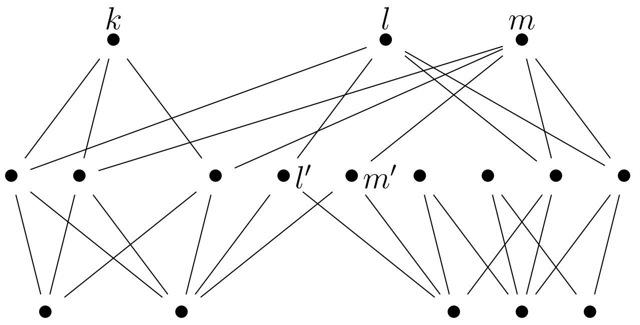


Figure 3.7: Making $\widehat{U}_l$ and $\widehat{U}_m$ connected by adding the points l' and m'

Then, Proposition 3.25 allows us to assume from now on in this section that the tilded minimal open sets of elements of height greater than 1 are connected. Now, once we have 1-h-regularized a given space, the subspace $\widehat{U}_x$ is an h-regular space of height 1, for every element x of height 2, therefore $\widehat{U}_x$ is weak homotopy equivalent to a finite wedge of 1-dimensional spheres by Lemma 3.4. We can *separate* such 1-spheres in order to satisfy the (h-regular) condition $\widehat{U}_x \overset{\text{we}}{\approx} S^1$ as it is shown in the next result.

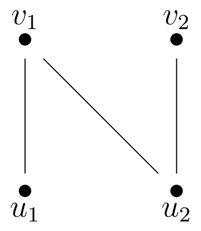
Proposition 3.26. *Let X be a 1-h-regular finite T_0-space and let $x \in X$ such that $h(x) = 2$. Suppose $\widehat{U}_x^X \overset{we}{\approx} \bigvee_{i=1}^q S^1$ for $q > 1$. Then there exists a 1-h-regular finite T_0-space X_1 such that $\widehat{U}_x^{X_1} \overset{we}{\approx} \bigvee_{i=1}^{q-1} S^1$ and $X \approx_2 X_1$.*

Proof. Let $\#\text{mnl}(\widehat{U}_x^X) = r$ and $\#\text{mxl}(\widehat{U}_x^X) = R$. Observe that $R = r + q - 1 > r$ by Lemma 3.4. Let M be a core of $\widehat{U}_x^X$, then M is connected and $M \overset{\text{we}}{\approx} \bigvee_{i=1}^q S^1$. Also, by Lemmas 3.2 and 3.3, M is h-regular. Note that $h(M) = 1$, since $h(x) = 2$. We divide the proof of the statement in three steps:

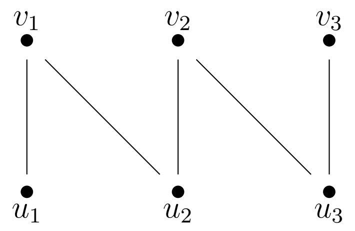
STEP 1: *Searching a $2s$-crown in M.* We will show that there exists, for some $s > 1$, a $2s$-crown $\{u_1, \dots, u_s, v_1, \dots, v_s\}$, where $\{u_1, \dots, u_s\} \subseteq \text{mnl}(M)$ and $\{v_1, \dots, v_s\} \subset$

$\mathrm{mxl}(M)$.

Take $u_1 \in \mathrm{mnl}(M)$ and $v_1 \in \mathrm{mxl}(M)$ such that $u_1 < v_1$. By the h-regularity of M, there exists $u_2 \in \mathrm{mnl}(M) - \{u_1\}$ such that $\widehat{U}^M_{v_1} = \{u_1, u_2\}$. Since M is a core, u_2 is not a beat point of M then there exists $v_2 \in \mathrm{mxl}(M) - \{v_1\}$ such that $u_2 < v_2$.



By the h-regularity of M, there exists $u_3 \in \mathrm{mnl}(M) - \{u_2\}$ such that $\widehat{U}^M_{v_2} = \{u_2, u_3\}$. Here we have two cases: if $u_3 = u_1$ then $\{u_1, u_2, v_1, v_2\}$ is a 4-crown in M; otherwise, since u_3 is not a beat point of M, there exists $v_3 \in \mathrm{mxl}(M) - \{v_2\}$ such that $u_3 < v_3$:



Continuing with this procedure, suppose that at some point we have found a subspace of M looking like the Figure 3.8, where $2 < m < r$.

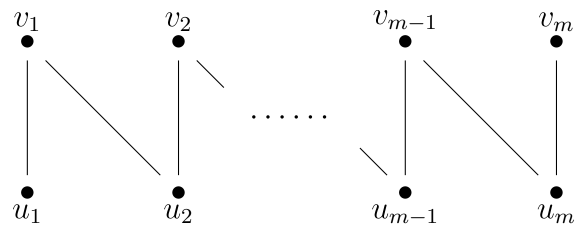


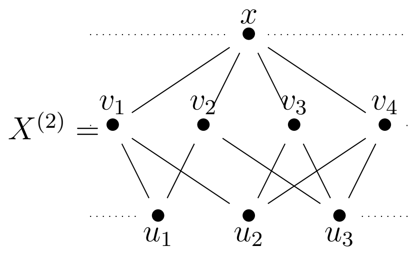
Figure 3.8: Step m in the construction of a $2s$-crown in M.

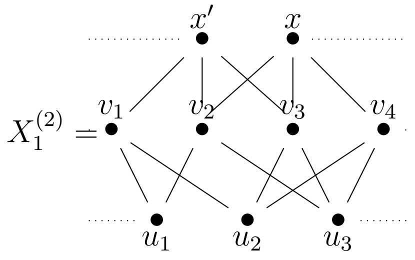
Then, since M is h-regular, there exists $u_{m+1} \in \mathrm{mnl}(M) - \{u_m\}$ such that $u_{m+1} < v_m$. If $u_{m+1} = u_k$ for some $k = 1, \dots, m-1$ then $\{u_k, u_{k+1}, \dots, u_m, v_k, v_{k+1}, \dots, v_m\}$ is a $2(m-k+1)$-crown in M; otherwise there exists $v_{m+1} \in \mathrm{mxl}(M) - \{v_m\}$ such that $u_{m+1} < v_{m+1}$.

This process must finish because the *worst* case is when $m = \#\mathrm{mnl}(M)$ in Figure 3.8. In this case, by the h-regularity of M, there exists necessarily some $k < m$ such that $u_k \in \mathrm{mnl}(M)$ and $u_k < v_m$, obtaining the respective $2(m-k+1)$-crown. Therefore, relabeling the points in the found crown if it is necessary, we have obtained a $2s$-crown $\{u_1, \ldots, u_s, v_1, \ldots, v_s\}$, where $r \geqslant s > 1$, $\{u_1, \ldots, u_s\} \subseteq \mathrm{mnl}(M)$ and $\{v_1, \ldots, v_s\} \subset \mathrm{mxl}(M)$.

STEP 2: *Construction of* X_1. Bearing in mind STEP 1, we label the rest of minimal and maximal elements of $\widehat{U}_x^X$ in such a way that $\mathrm{mnl}(\widehat{U}_x^X) = \{u_1, \ldots, u_s, u_{s+1}, \ldots, u_r\}$ and $\mathrm{mxl}(\widehat{U}_x^X) = \{v_1, \ldots, v_s, v_{s+1}, \ldots, v_R\}$. Consider the space $X_1 = X \cup \{x'\}$, with $x' \notin X$, whose minimal basis $\{U_z^{X_1}\}_{z \in X_1}$ satisfies:

$$\widehat{U}_z^{X_1} = \begin{cases} \{u_1, \ldots, u_s, v_1, \ldots, v_s\} & , \ z = x' \\ \{u_1, \ldots, u_r, v_2, \ldots, v_R\} & , \ z = x \\ \widehat{U}_z^X \cup \{x'\} & , \ z \in \widehat{F}_x^X \\ \widehat{U}_z^X & , \ z \notin \widehat{F}_x^X . \end{cases}$$





By construction of X_1, $\{x, x'\}$ is an antichain of X_1 such that:

$$z \in \widehat{F}_x^{X_1} \Longleftrightarrow x \in \widehat{U}_z^{X_1} \Longleftrightarrow z \in \widehat{F}_x^X ,$$

$$z \in \widehat{F}_{x'}^{X_1} \Longleftrightarrow x' \in \widehat{U}_z^{X_1} \Longleftrightarrow z \in \widehat{F}_x^X ,$$

hence $\widehat{F}_x^X = \widehat{F}_x^{X_1} = \widehat{F}_{x'}^{X_1}$. Also, by Proposition 3.13, the subposet $U_{x'}^{X_1} \cup U_x^{X_1}$ is contractible because $U_{x'}^{X_1} \cap U_x^{X_1} = \widehat{U}_{x'}^{X_1} \cap \widehat{U}_x^{X_1} = \{u_1, \ldots, u_s, v_2, \ldots, v_s\}$ is connected by Lemma 3.7 and

$$\begin{aligned} \#\mathrm{mxl}\left(\widehat{U}_{x'}^{X_1} \cap \widehat{U}_x^{X_1}\right) &= s - 1 \\ &= \#\mathrm{mnl}\left(\widehat{U}_{x'}^{X_1} \cap \widehat{U}_x^{X_1}\right) - 1 \Longrightarrow \widehat{U}_{x'}^{X_1} \cap \widehat{U}_x^{X_1} \overset{\text{we}}{\approx} *. \end{aligned}$$

Therefore $X_1 \nearrow\searrow X$ by Lemma 3.18.

STEP 3: *End of the proof.* Note that, since $\widehat{U}_{x'}^{X_1} = \{u_1, \ldots, u_s, v_1, \ldots, v_s\}$ is a $2s$-crown, $\widehat{U}_{x'}^{X_1} \overset{\text{we}}{\approx} S^1$ (by using Lemma 3.4). On the other hand, taking $Z = \widehat{U}_x^X$, $Y = \widehat{U}_{x'}^{X_1}$ and $x = v_1$ in Lemma 3.7, we obtain the conectedness of $\widehat{U}_x^{X_1} = \{u_1, \ldots, u_r, v_2, \ldots, v_R\}$ and therefore, by Lemma 3.4

$$\begin{aligned}\#\mathrm{mxl}\left(\widehat{U}_x^{X_1}\right) &= R-1\\ &= (r+q-1)-1\\ &= \#\mathrm{mnl}\left(\widehat{U}_x^{X_1}\right)+(q-1)-1 \Longrightarrow \widehat{U}_x^{X_1} \overset{\text{we}}{\approx} \bigvee_{i=1}^{q-1} S^1.\end{aligned}$$

Finally, observe that $W := X - X^{(2)}$ equals $X_1 - X_1^{(2)}$ as sets; moreover, for all $y \in W$, since $x' \notin W$ then $U_y^W = U_y^X \cap W = U_y^{X_1} \cap W$, hence the topologies of X and X_1 coincide on W. ∎

By iterating the method described in Proposition 3.26, the following result is obtained.

Corollary 3.27. *Let X be a 1-h-regular finite T_0-space and let $x \in X$ such that $h(x) = 2$ and $\widehat{U}_x^X \overset{we}{\approx} \bigvee_{i=1}^q S^1$ for $q > 1$. Then there exists a 1-h-regular finite T_0-space X_x such that $\widehat{U}_x^{X_x} \overset{we}{\approx} S^1$ and $X \approx_2 X_x$.*

Proof. By applying Proposition 3.26 succesive times, we can construct a sequence of spaces

$$X = X_q \subset X_{q-1} \subset \cdots \subset X_1$$

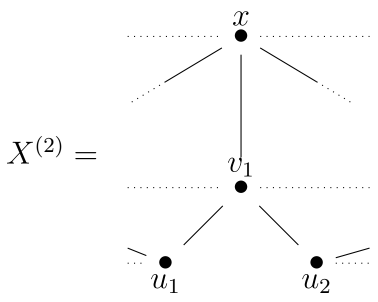
such that for each $k = 1, \ldots, q-1$ we have $\widehat{U}_x^{X_k} \overset{\text{we}}{\approx} \bigvee_{i=1}^k S^1$ and $X_{k+1} \approx_2 X_k$. Taking $X_x = X_1$ the result is obtained. ∎

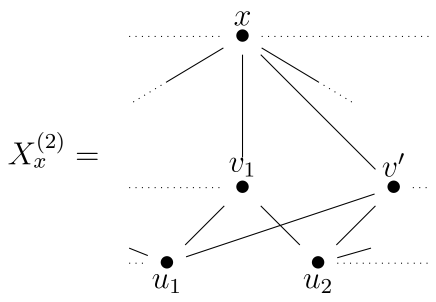
In case the subspace $\widehat{U}_x^X$ is contractible, the element x is a weak point and we could delete it from the space without risk of modifying its weak homotopy type. However, we want to show a different alternative in order to satisfy $\widehat{U}_x^{X_x} \overset{\text{we}}{\approx} S^1$.

Lemma 3.28. *Let X be a 1-h-regular finite T_0-space and let $x \in X$ such that $h(x) = 2$ and $\widehat{U}_x^X$ is contractible. Then there exists a 1-h-regular finite T_0-space X_x such that $\widehat{U}_x^{X_x} \overset{we}{\approx} S^1$ and $X \approx_2 X_x$.*

Proof. Note that, by the h-regularity of $\widehat{U}_x^X$, there exist $u_1, u_2, v_1 \in \widehat{U}_x^X$ such that $u_1, u_2 < v_1$. On the other hand, by Lemma 3.4, $\#\mathrm{mxl}(\widehat{U}_x^X) = \#\mathrm{mnl}(\widehat{U}_x^X) - 1$. Consider the space $X_x = X \cup \{v'\}$, with $v' \notin X$, whose minimal basis $\{U_z^{X_x}\}_{z \in X_x}$ satisfies:

$$\widehat{U}_z^{X_x} = \begin{cases} \{u_1, u_2\} & , \ z = v' \\ \widehat{U}_z^X \cup \{v'\} & , \ z \in F_x^X \\ \widehat{U}_z^X & , \ z \notin F_x^X . \end{cases}$$





Observe that $z \in \widehat{F}^{X_x}_{v'} \iff v' \in \widehat{U}^{X_x}_z \iff z \in F^X_x$, then $\widehat{F}^{X_x}_{v'} = F^X_x$ and hence v' is a beat point of X_x. Since $\widehat{U}^X_x$ is h-regular connected, $\widehat{U}^{X_x}_x$ is also h-regular connected and therefore we can apply Lemma 3.4 to obtain:

$$\begin{aligned}\#\text{mxl}(\widehat{U}^{X_x}_x) &= \text{mxl}(\widehat{U}^X_x \cup \{v'\}) \\ &= \#\text{mxl}(\widehat{U}^X_x) + 1 \\ &= \#\text{mnl}(\widehat{U}^X_x) \\ &= \#\text{mnl}(\widehat{U}^{X_x}_x) \Longrightarrow \widehat{U}^{X_x}_x \overset{\text{we}}{\approx} S^1.\end{aligned}$$

Finally, observe that $W := X - X^{(2)}$ equals $X_x - X^{(2)}_x$ as sets; moreover, for all $y \in W$, since $v' \notin W$ then $U^W_y = U^X_y \cap W = U^{X_x}_y \cap W$. Then, the topologies of X and X_x coincide on W. ■

Theorem 3.29. *Let X be a finite T_0-space without beat points. There exists a 2-h-regular finite T_0-space L such that $X \approx_2 L$.*

Proof. If X is 2-h-regular, take $L = X$; otherwise consider M a 1-h-regular finite T_0-space without beat points such that $X \approx_1 M$ (by Theorem 3.24).

Let $X^{(2)} - X^{(1)} = M^{(2)} - M^{(1)} = \{x_1, \dots, x_m\}$. For x_1, consider the following cases:

- If $\widehat{U}^M_{x_1}$ is connected, take $M_1 = M$.
- If $\widehat{U}^M_{x_1}$ is not connected, take a 1-h-regular finite T_0-space M_1 such that $\widehat{U}^{M_1}_{x_1}$ is connected and $M \approx_1 M_1$ (by Proposition 3.25).

In any case, we have that $M \approx_1 M_1$ and $\widehat{U}^{M_1}_{x_1}$ is a 1-h-regular connected finite T_0-space of height 1, hence $\widehat{U}^{M_1}_{x_1} \overset{\text{we}}{\approx} \bigvee_{i=1}^q S^1$ for some $q \in \mathbb{N}_0$.

- If $q = 1$, $\widehat{U}^{M_1}_{x_1} \overset{\text{we}}{\approx} S^1$ so that we take $M_{1,q} = M_1$.
- If $q = 0$, by Lemma 3.28, there exists a 1-h-regular finite T_0-space $M_{1,q}$ such that $\widehat{U}^{M_{1,q}}_{x_1} \overset{\text{we}}{\approx} S^1$ and $M_1 \approx_2 M_{1,q}$.
- If $q > 1$, by Corollary 3.27, there exists a 1-h-regular finite T_0-space $M_{1,q}$ such that $\widehat{U}^{M_{1,q}}_{x_1} \overset{\text{we}}{\approx} S^1$ and $M_1 \approx_2 M_{1,q}$.

In any case, we have that $\widehat{U}_{x_1}^{M_1} \overset{\text{we}}{\approx} S^1$ and $M_1 \approx_2 M_{1,q}$.

By repeating this process for $x_2, \dots, x_m$, we obtain a sequence of 1-h-regular finite T_0-spaces

$$M = M_{0,q} \subset M_{1,q} \subset \cdots \subset M_{m,q}$$

such that for each $k = 0, \dots, m-1$ we have $\widehat{U}_{x_i}^{M_{k,q}} \overset{\text{we}}{\approx} S^1$ for all $i = 1, \dots, k$ and $M_{k,q} \approx_2 M_{k+1,q}$. Taking $L = M_{m,q}$ the result is obtained. ■

Corollary 3.30. *Let X be a finite T_0-space of height at most 2. Then there exists an h-regular finite T_0-space L such that $X \nearrow\searrow L$.*

Proof. Apply Theorem 3.29 to a core of X. ■

Continuing with our example, in order to 2-h-regularize the space in Figure 3.7, we consider the elements of height 2. Note that $\widehat{U}_m$ has 5 minimal elements and 5 maximal elements so that by Lemma 3.4, such subspace is weak homotopy equivalent to S^1, satisfying the h-regular property. On the other hand, $\widehat{U}_k$ is a finite model of $S^1 \vee S^1$ and $\widehat{U}_l$ is contractible. Then, we use Corollary 3.27 and Lemma 3.28 in order to obtain the h-regular space in Figure 3.9.

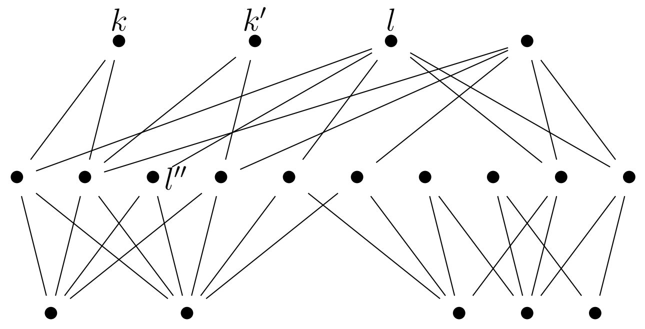


Figure 3.9: 2-h-regularization of M.

The statement of Corollary 3.30 is not surprising in a theoretical sense because we already know that the barycentric subdivision X' satisfies such characteristics. However, our construction permits to obtain from X an h-regular space smaller than X', so that we can deal with bigger finite spaces in order to compute homotopical invariants as we will show in Section 3.4. For example, observe that the space in Figure 3.9 is an h-regular space with 19 elements simple homotopy equivalent to M; in contrast, the barycentric subdivision of M has 63 elements.

The method of h-regularization has been completely developed for finite T_0-spaces of height at most 2. The complete process can be done for such spaces thanks to the fact that h-regular finite models of 1-spheres can be characterized by using Lemma 3.4. Unfortunately, there is not an analog result on finite models of spheres in greater dimensions. However, most of the results that have been shown in this chapter are applicable to minimal finite T_0-spaces with no constraint on their heights. In particular, Theorems 3.24 and 3.29 can be used as first steps in the search of h-regular spaces weak homotopy equivalent to finite T_0-spaces of height greater than two, as an alternative to the construction of barycentric subdivisions.

3.4 Examples and computations

In Section 3.3 we have shown a procedure to obtain a simple homotopy equivalent h-regular space to a given one of height at most 2. We have implemented this procedure in Kenzo in order to be incorporated to our new module devoted to computations on finite topological spaces. In this section we will show some examples of computations by using the new functions in Kenzo.

3.4.1 Implemented functions for h-regularization

First of all, we present some of the functions related with the h-regularization process that we have developed.

`separate-segments` *minimal-finspace point*
: It returns the `FINITE-SPACE` X_x given in Corollary 3.22 i.e. it separates the edges of *minimal-finspace* whose head is *point*. The parameter *minimal-finspace* must be a `FINITE-SPACE` without beat points and *point* must be an element of height 1 such that $\#\widehat{U}_{point} \geqslant 3$.

`1-h-regularization` *minimal-finspace*
: This function returns the `FINITE-SPACE` L given in Theorem 3.24 i.e. the 1-h-regularization of *minimal-finspace*. The parameter *minimal-finspace* must be an instance of `FINITE-SPACE` without beat points.

`Un-connect` *1hreg-finspace point*
: It returns the `FINITE-SPACE` X_1 given in Proposition 3.25 i.e. it makes the subspace $\widehat{U}^{X_1}_{point}$ connected. The parameter *1hreg-finspace* must be a 1-h-regular `FINITE-SPACE` of height 2 without beat points and *point* must be an element of height 2.

`Un-connected` *1hreg-finspace*
: It returns the `FINITE-SPACE` X_1 given in Proposition 3.25 applied to all the elements of height 2 i.e. it makes $\widehat{U}^{X_1}_{x}$ connected for every x of height 2. The parameter *1hreg-finspace* must be a 1-h-regular `FINITE-SPACE` of height 2 without beat points.

`find-crown` *minimal-finspace*
: It returns a list of elements which made part of a crown contained in *minimal-finspace* (see STEP 1 in the proof of Proposition 3.26). The parameter *1hreg-finspace* must be a 1-h-regular `FINITE-SPACE` of height 1 without beat points.

`separate-cycles` *1hreg-finspace point*

It returns the `FINITE-SPACE` X_x given in Corollary 3.27 i.e. it makes $\widehat{U}^{X_x}_{point}$ weak homotopy equivalent to S^1. The parameter *1hreg-finspace* must be a 1-h-regular `FINITE-SPACE` of height 2 without beat points and *point* must be an element of height 2.

`2-h-regularization` *minimal-finspace*

This function returns the `FINITE-SPACE` L given in Theorem 3.29 i.e. the 2-h-regularization of *minimal-finspace*. The parameter *minimal-finspace* must be an instance of `FINITE-SPACE` without beat points.

For testing our procedure, auxiliary functions to create `FINITE-SPACES` of height at most 2 were also implemented:

`wedge-spheres` *num-minimals num-spheres*

It returns a topogenous matrix of type `matrice` representing a finite T_0-space X weak homotopy equivalent to a wedge of *num-spheres* 1-dimensional spheres, such that $h(X) = 1$ and $\#\text{mnl}(X) =$ *num-minimals*. The parameters must satisfy the conditions *num-minimals* $\geqslant 2$ and *num-spheres* $\geqslant 1$.

`random-2space` *dimension*

This function returns a random minimal `FINITE-SPACE` of cardinality *dimension*, height 2 and without beat points. The parameter *dimension* must satisfy the condition *dimension* $\geqslant 6$.

3.4.2 h-regularizing finite spaces in Kenzo

Consider the finite T_0-space in Figure 3.10, which is a space weak homotopy equivalent to a wedge of two 1-spheres i.e. $S^1 \vee S^1$. We represent it in Kenzo by `wedge42`. We can identify a crown in this space by using the function `find-crown`:

```
> (setf wedge42 (build-finite-space :top (wedge-spheres 4 2)
                                    :orgn '(wedge 4 2)))
[K1 Finite-Space]
> (find-crown wedge42)
(7 8 6 5)
```

The obtained result `(7 8 6 5)` indicates that the subspace $U_5 \cup U_6 \cup U_7 \cup U_8$ is an 8-crown contained in `wedge42`. Now, we consider the non-Hausdorff suspension of `wedge42`, which is denoted by `nhs-wedge42`:

```
> (setf nhs-wedge42 (non-hausdorff-suspension wedge42))
[K2 Finite-Space]
```

Note that `nhs-wedge42` is 1-h-regular but it is not h-regular: each of the new two maximal points satisfy $\widehat{U}_x \overset{\text{we}}{\approx} S^1 \vee S^1$. Calling the slot `:stong` for `nhs-wedge42` and for `(separate-cycles nhs-wedge42 10)` we obtain the Stong matrices, respectively:

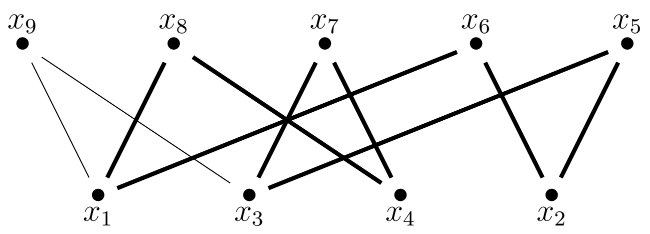


Figure 3.10: The space `wedge42` (an 8-crown contained in `wedge42` is highlighted).

$$S_X = \left[\begin{array}{ccccccccccc}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & \mathbf{0} & 0 \\
 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \mathbf{0} & 0 \\
 & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \mathbf{0} & 0 \\
 & & & 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} & 0 \\
 & & & & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 \\
 & & & & & 1 & 0 & 0 & 0 & \mathbf{1} & 1 \\
 & & & & & & 1 & 0 & 0 & \mathbf{1} & 1 \\
 & & & & & & & 1 & 0 & \mathbf{1} & 1 \\
 & & & & & & & & 1 & \mathbf{1} & 1 \\
 & & & & & & & & & 1 & 0 \\
 & & & & & & & & & & 1
\end{array}\right], \quad S_Y = \left[\begin{array}{cccccccccccc}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & \mathbf{0} & \mathbf{0} & 0 \\
 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & 0 \\
 & & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \mathbf{0} & \mathbf{0} & 0 \\
 & & & 1 & 0 & 0 & 1 & 1 & 0 & \mathbf{0} & \mathbf{0} & 0 \\
 & & & & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 1 \\
 & & & & & 1 & 0 & 0 & 0 & \mathbf{1} & \mathbf{1} & 1 \\
 & & & & & & 1 & 0 & 0 & \mathbf{1} & \mathbf{0} & 1 \\
 & & & & & & & 1 & 0 & \mathbf{1} & \mathbf{1} & 1 \\
 & & & & & & & & 1 & \mathbf{0} & \mathbf{1} & 1 \\
 & & & & & & & & & 1 & \mathbf{0} & 0 \\
 & & & & & & & & & & 1 & 0 \\
 & & & & & & & & & & & 1
\end{array}\right].$$

The column 10 in S_X was *separated* in the two columns in bold in S_Y. Note that the nonzero entries in column 10 of S_Y correspond to the indexes in the list `(find-crown wedge42)`, while the nonzero entries in column 11 represent another cycle contained in `wedge42`: $U_5 \cup U_6 \cup U_8 \cup U_9$ (this cycle is not a crown, but its core $U_5 \cup U_6 \cup U_9$ is a 6-crown). The same separation can be done to the last column of S_Y. Above computations are involved in the process of computing `(2-h-regularization nhs-wedge42)`. The h-regularization of `nhs-wedge42` is shown in Figure 3.11. Note that the cardinality of the barycentric subdivision `nhs-wedge42` is greater than the cardinality of its h-regularization:

```
> (2-h-regularization nhs-wedge42)
[K5 Finite-Space]
> (cardinality (k 5))
13
> (bar-subdivision nhs-wedge42)
[K6 Finite-Space]
> (cardinality (k 6))
59
```

Therefore, we can use the functions described in Subsection 2.2.4.1 to compute the homology of `nhs-wedge42` by using its h-regularization:

```
> (h-regular-homology (k 5) 0)
```

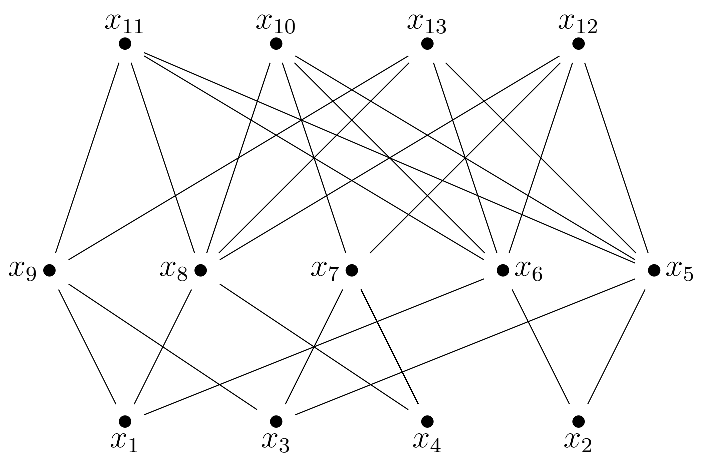


Figure 3.11: The h-regularization of `nhs-wedge42`.

```
Component Z
> (h-regular-homology (k 5) 1)

> (h-regular-homology (k 5) 2)
Component Z
Component Z
```

3.4.3 Generating random examples and comparison of the sizes of X_h and X'

We have observed in the examples shown along the chapter, concretely the space in Figure 3.5 in Section 3.3 and the space `nhs-wedge42` in Subsection 3.4.2, that the number of points of our h-regularization method is smaller than the cardinality of the respective barycentric subdivisions.

By using our implementation of the method in Kenzo, we have tested the h-regularization of random finite T_0-spaces. We have constructed 20 arbitrary finite T_0-spaces X without beat points by using the function `random-2space` for some dimensions ($\#X$). Then, we have used the method `2-h-regularization` in order to obtain instances of the class `FINITE-SPACE` representing the h-regularizations (X_h) of our testing examples; in the same way, the function `bar-subdivision` is used to compute the barycentric subdivisions (X'). Once we have computed those spaces, the average of the cardinalities of the spaces X, X_h and X' are found in order to compare the results obtained. In Table 3.1 we have summarized this information.

To point out the benefits of the method, we present the following computations. The h-regularization method can be used to *repair* some perturbations applied to finite spaces coming from the simplicial complex world. For example, we have imported in Kenzo the

Table 3.1: Comparison of the cardinalities of the barycentric subdivision (X') and our method of h-regularization (X_h). In each row, 20 random spaces of size $\#X$ and height 2, without beat points, have been computed.

$\#X$	Average $\#X_h$	Average $\#X'$	Average % reduction
10	22.30	73.10	69.66
15	67.45	177.75	62.40
20	144.15	325.55	56.11
25	257.50	539.50	53.87
30	435.20	847.20	50.19
35	731.55	1266.75	43.14

data of the facets of a random 2-dimensional simplicial complex with 25 vertices and 751 triangles, picked with probability 0.328, as described in the example *"rand2_n25_p0.328"* in [BL17], and we have constructed its face poset `K1`.

```
> (setf data_folder "...")
> (setf K1 (import-facets-to-finite-space "rand2_n25_p0.328"))
[K1 Finite-Space]
```

Then, we have taken a wedge of four copies of this finite T_0-space and a `random-2space` of 19 elements (a non-h-regular perturbation).

```
> (setf F2 (random-2space 19))
[K2 Finite-Space]
> (setf W (wedge-at-x1 K1 K1 K1 K1 F2))
[K3 Finite-Space]
> (cardinality (k 3))
4373
```

As we can observe, the resulting space has 4373 points. Kenzo needed around two minutes to compute the h-regularization of `W` and its homology groups, while the barycentric subdivision of `W` could not be constructed.

```
> (time (2-h-regularization W))
Timing the evaluation of (2-H-REGULARIZATION W)

User time    =  0:01:04.046
System time  =        0.187
Elapsed time =  0:01:04.123
[K4 Finite-Space]

> (time (h-regular-homology-sim (k 4)))
```

```
Timing the evaluation of (H-REGULAR-HOMOLOGY-SIM (K 4))

Homology in dimension 0 : Z
Homology in dimension 1 :
Homology in dimension 2 :  Z ^ 1954
User time    =       57.015
System time  =        0.078
Elapsed time =       57.257
```

Chapter 4

Integration of Kenzo and SageMath

The algorithms and methods described in preceding chapters have been implemented in the Kenzo program. As has been pointed out, this is a powerful software that has obtained some results which had never been determined before [RS06] and, as far as we know, it is the only program able to compute homology of infinite structures using effective homology and to compute algorithmically homotopy groups combining the Whitehead tower method [Whi52] and the effective homology technique. Despite these capabilities, Kenzo lacks of a friendly interface and its use demands some knowledge on the Common Lisp programming language, which can be considered tedious (indeed, the last official Kenzo release is not fully compatible with all the Common Lisp dialects). Moreover, there exist additional modules for computing other algebraic topology structures (for example, the module implementing discrete vector fields that has been used in Chapter 2), but they are available only in the personal websites of the authors.

Meanwhile, SageMath [Dev20] is a general purpose computer algebra system. It is one of the most used programs in both education and research environments, in a wide range of mathematical branches. It possesses a friendly graphical user interface, based on Jupyter notebooks, and it is mainly developed in Python.

Then, the integration of these systems is of benefit for both Kenzo and SageMath computations. In fact, there already exists a way to communicate Kenzo and SageMath, but it is restricted to some particular cases and is based on external Kenzo files which have to be loaded manually in SageMath (see [Pal20]), hence there is no proper connection between both systems.

In this chapter, we will describe the current version of an interface between Kenzo and SageMath that we have developed in a joint work with Jose Divasón and Miguel Marco-Buzunáriz. Moreover, we will show a new module that we have created for finite topological spaces in SageMath, allowing to integrate the methods we have described in above chapters, by means of the developed interface.

The work explained in this chapter has been presented in [CRMBR19] and [CRDMBR19] (distinguished with the *Best Software Demonstration Award*) and the preprint [CRDMBR20].

4.1 SageMath

SageMath [Dev20] is a free open-source mathematics software system built on top of nearly 100 open-source packages (such as NumPy, SciPy, matplotlib, Sympy, Maxima, FLINT, GAP, R), featuring a unified and friendly graphical user interface based on Jupyter notebooks and mainly developed in Python. It was created by William Stein in 2005 with the mission statement of *creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab* and following the philosophy of *building the car instead of reinventing the wheel*, in the sense that it includes many other libraries and systems developed for specific purposes, and makes them work together under a common abstraction layer. Since its creation, SageMath has grown to a big international project, with hundreds of collaborators all around the world, developing new versions released every few months, with bugfixes and new functionalities.

Almost all areas of mathematics are represented in SageMath, at various levels of sophistication. Symbolic calculus, 2D and 3D graphics, polynomials, graph theory, group theory, abstract algebra, algebraic topology, combinatorics, cryptography, elliptic curves and modular forms, numerical mathematics, linear algebra and matrix calculations, are some of the fields that are included in the scope of SageMath, which also expand into neighboring fields like Statistics and Physics.

The SageMath development process is carried out by means of an issue tracking system called Trac. A ***ticket*** is the name given for an item on the server, where anyone can post on the Trac server: to report a bug, to submit new code, to review code which has not been yet included in SageMath or to suggest some corrections for the documentation.

4.1.1 Algebraic topology in SageMath

SageMath provides some tools for working in algebraic topology. Chain complexes and their homology as well as computations over objects of finite type, such as simplicial complexes, Δ-complexes, cubical complexes, simplicial sets and a class of generic cell complexes, are also available.

4.1.1.1 Chain complexes

The implementation built in SageMath for chain complexes represents a more general notion than the given in Definition 1.53. A *chain complex* $C_* = (C_i, d_i)$ in SageMath is a sequence of free R-modules over a conmmutative ring R, indexed by any free abelian group F, and R-module maps $d_i : C_i \longrightarrow C_{i+r}$, for a fixed $r \in F$ (called *degree of differentials*), such that $d_{i+r}d_i = 0$ for all $i \in F$. However, homology calculations in chain complexes will only work when the ring R is either $\mathbb{Z}$ or a field. For this reason, we will consider chain complexes in SageMath with $F = R = \mathbb{Z}$ (then, the C_i's are free abelian groups) and degree of differentials $r = -1$ (as in Definition 1.53).

Let us show some examples of contructions and computations on chain complexes in SageMath. Consider the oriented simplicial complex K in Figure 4.1 (this is the same complex in Figure 1.8). We will construct the chain complex $C_*(K)$ which generates the simplicial homology groups of K.

$$K = \{[a],[b],[c],[d],[e],\\ [a,b],[a,c],[a,d],\\ [b,c],[c,d],[c,e],\\ [d,e],[a,b,c]\}$$

$|K|$

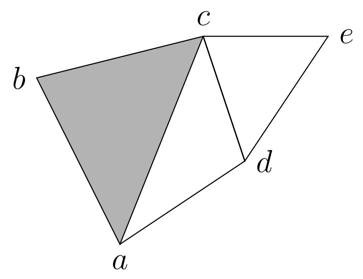


Figure 4.1: A simplicial complex K and its geometric realization $|K|$.

In SageMath, there are various admissible types of data to define a chain complex, but all of them need as inputs the matricial representation of the nonzero differential maps d_p.

The matrices representing the nonzero differential maps of $C_*(K)$, as defined by (1.7), are the following:

$$d_1 = \begin{array}{c} \\ [a] \\ [b] \\ [c] \\ [d] \\ [e] \end{array} \begin{array}{c} \begin{array}{ccccccc} [a,b] & [a,c] & [a,d] & [b,c] & [c,d] & [c,e] & [d,e] \end{array} \\ \begin{pmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}, \quad d_2 = \begin{array}{c} \\ [a,b] \\ [a,c] \\ [a,d] \\ [b,c] \\ [c,d] \\ [c,e] \\ [d,e] \end{array} \begin{array}{c} [a,b,c] \\ \begin{pmatrix} 1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{array}$$

Above matrices can be constructed in SageMath by means of a *dictionary* whose keys are the indexes of the non-zero entries:

```
sage: d1_dict = {(0,0):-1, (1,0):1, (0,1):-1, (2,1):1,
                 (0,2):-1, (3,2):1, (1,3):-1, (2,3):1,
                 (2,4):-1, (3,4):1, (2,5):-1, (4,5):1,
                 (3,6):-1, (4,6):1}
sage: d2_dict = {(0,0):1, (1,0):-1, (3,0):1}
sage: d1 = matrix(5, 7, d1_dict)
sage: d2 = matrix(7, 1, d2_dict)
```

One way to define a chain complex in SageMath is by means of a dictionary with integers for keys, and with values the matrices representing the differentials coming from the respective key (as we pointed out at the beginning of the subsection, the degree of the differentials must be specified to be -1). In our case, the chain complex is constructed by:

```
sage: C = ChainComplex({1:d1, 2:d2},
                       degree_of_differential=-1); C
Chain complex with at most 3 nonzero terms over Integer Ring
sage: ascii_art(C)
                                                  [ 1]
                                                  [-1]
            [-1 -1 -1  0  0  0  0]                [ 0]
            [ 1  0  0 -1  0  0  0]                [ 1]
            [ 0  1  0  1 -1 -1  0]                [ 0]
            [ 0  0  1  0  1  0 -1]                [ 0]
            [ 0  0  0  0  0  1  1]                [ 0]
 0 <-- C_0 <----------------------- C_1 <----- C_2 <-- 0
```

Note that the base ring was not specified in the definition of `C`. The system examines the input matrices to determine a ring over which they are all naturally defined, and this becomes the base ring for the complex (in our case, the integer ring $\mathbb{Z}$ was chosen).

Once a chain complex is created, its homology can be obtained. Then, the simplicial homology groups of K are given:

```
sage: C.homology()
{0: Z, 1: Z x Z, 2: 0}
```

The generators of the non-trivial homology groups (see Figure 4.2), can be computed as follows:

```
sage: C.homology(0, generators=True)
[(Z, Chain(0:(0, 1, 0, 0, 0)))]
sage: C.homology(1, generators=True)[0]
(Z, Chain(1:(0, 1, -1, 0, 0, 1, -1)))
sage: C.homology(1, generators=True)[1]
(Z, Chain(1:(0, 0, 0, 0, 1, -1, 1)))
```

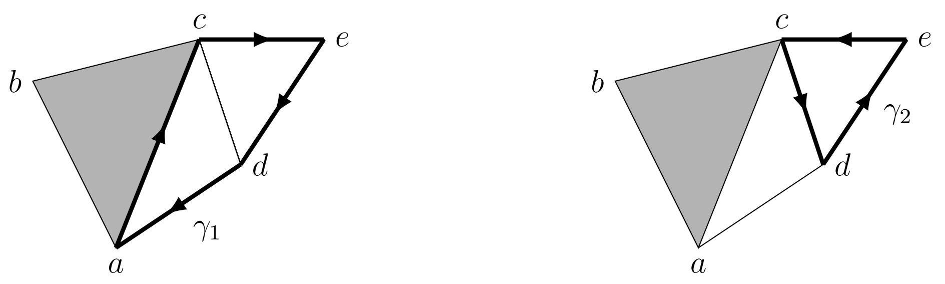


Figure 4.2: Representation of the generators of $H_1(K) = \langle\gamma_1\rangle \oplus \langle\gamma_2\rangle \cong \mathbb{Z} \oplus \mathbb{Z}$ computed in SageMath.

4.1.1.2 Finite simplicial complexes

Finite simplicial complexes in SageMath are defined by means of the set of its facets (maximal faces). This parameter can be specified by either a list, a tuple or a set whose elements are lists (tuples or sets) of vertices representing the facets, or by a pair `(f,X)` where `X` is the set of points and `f` is a boolean function that determines if a list of elements from `X` is a simplex or not.

Note that the facets of K are $[a, b, c]$, $[a, d]$, $[c, d]$, $[c, e]$ and $[d, e]$. This can be used as the input data for constructing K in SageMath:

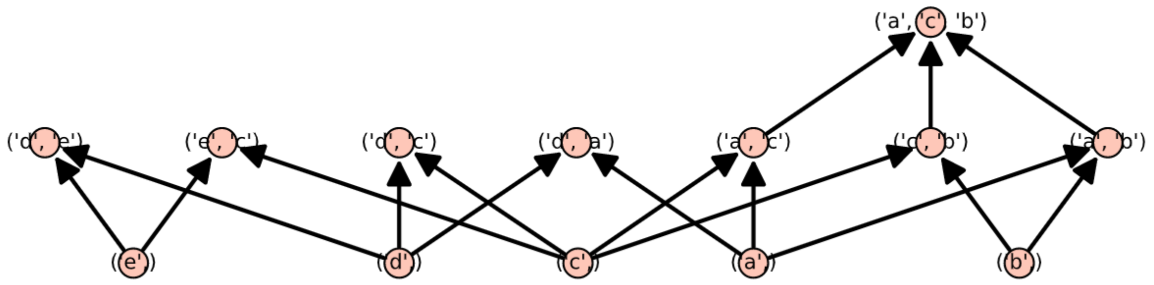
```
sage: facets_K = [['a','b','c'],['a','d'],['c','d'],
                  ['c','e'],['d','e']]
sage: K = SimplicialComplex(facets_K); K
Simplicial complex with vertex set ('a', 'b', 'c', 'd', 'e')
and 5 facets
```

The `homology` method for simplicial complexes computes the reduced homology. Applying this to `K` we obtain:

```
sage: K.homology(0)
0
sage: K.homology(1)
Z x Z
sage: K.homology(2)
0
```

Apart from this direct construction method, Sage provides some constructions that can be used to build new simplicial complexes from a previously constructed one (Alexander dual, barycentric subdivision, stellar subdivision, cone, suspension), or from various simplicial complexes (connected sum, disjoint union, join, wedge, cartesian product). Moreover, the face poset of a simplicial complex can be constructed as an instance of the class `FinitePoset`:

```
sage: K.face_poset().show()
```



4.2 The SageMath-Kenzo interface

This section is divided in two parts. In the first one, we show the interface which has been developed to communicate Kenzo and SageMath. This interface allows to transform topological data structures from one system to the other one. In the second part, we present the integration of Kenzo into a particular (already existing) SageMath module. This integration allows that Kenzo calculations can be carried out from Sage in a more easy way.

4.2.1 Communication between SageMath and Kenzo and new classes in SageMath

In this section we present the (optional) package to install Kenzo within SageMath and the basics of the interface that connects both systems. This connection is made via the library interface to Embeddable Common Lisp (ECL) in SageMath.

Our Kenzo package for SageMath is based on the Kenzo version by Heber [Heb19], which is an adapted version of the original Kenzo, but including improvements to make it compatible with any Common Lisp system. We make it easy to install our Kenzo optional package in SageMath, simply by running:

```
sage -i kenzo
```

This command will download the source code of a suitable version of Kenzo that we maintain for this specific purpose [MB15], it will compile this code inside the ECL interpreter that SageMath includes, and it will install Kenzo. In fact, this can be an easy way to install Kenzo even if we just plan to use it as a standalone program. More concretely, we can use Kenzo with an ECL session via the command `sage -ecl`. This way, we could use Kenzo as usual, i.e., by means of Lisp commands.

This Kenzo optional package can also be used within a SageMath session via two main functions: `EclObject`, which takes a Python object and tries to find a Lisp representation for it, and `ecl_eval`, a function that evaluates a string in Lisp and returns the result. For instance, the following commands load Kenzo and define the Eilenberg–MacLane space $K(\mathbb{Z}, 3)$ by means of Lisp commands evaluated with `ecl_eval` in a SageMath session.

```
sage: from sage.libs.ecl import ecl_eval
sage: ecl_eval("(require :kenzo)")
sage: ecl_eval("(in-package #:cat)")
sage: ecl_eval("(setf K (k-z 3))")
<ECL: [K25 Abelian-Simplicial-Group]>
```

In the previous example, the execution is carried out in Kenzo via the ECL library, but there is no proper connection to SageMath classes. In addition, this embedding of Kenzo within SageMath does not ease its use. In fact, the same Lisp syntax and Kenzo specific commands must be used.

Thus, it is therefore appropriate to develop an interface to communicate both systems, in such a way that both systems can collaborate together in computations. The interface we have designed has been implemented in Python. The corresponding file has been included in the SageMath distribution. A ticket in Trac was created to this purpose:

`https://trac.sagemath.org/ticket/27880`

This interface automatically loads Kenzo inside the ECL library, which is itself loaded as a C-library in SageMath. This way, the communication between both systems is done via a C-library interface, which has a much lower overhead than other alternatives, such as pseudo-terminals or temporary files. Moreover, the interface provides functions to create Kenzo objects, and wrappers around them that allow to create new objects (with the corresponding wrapper) and call functions on them. When it makes sense, the output of those functions is converted to the corresponding SageMath object. The initial version of the interface was available in the SageMath distribution from version 8.7, including some basic functionalities. In particular, the interface exposes functions to create spheres, Eilenberg–MacLane spaces and Moore spaces. From these spaces, it allows the construction of cartesian products, classifying spaces, loop spaces, suspensions and tensor products. In addition, for each resulting space, the `homology` method can construct the corresponding homology groups.

From SageMath version 9.2, a second version of the package and the interface has been enhanced by adding some new functionalities. For instance, we have included some existing external modules of Kenzo that allow computing homotopy groups of 1-reduced simplicial sets (by using the Whitehead tower method) and some spectral sequences. In addition, we have developed methods for translating objects such as chain complexes, chain complex morphisms and simplicial sets both from SageMath to Kenzo and vice versa. In this way, we can build simplicial objects in SageMath, translate them into Kenzo and compute their homology and homotopy groups. This second version also updates the Kenzo package to fix bugs detected in an external module, to include a new algorithm to compute homotopy groups of not 1-reduced simplicial sets and other features.

We provide a website which allows the user to carry out online executions of commands and explore the obtained outputs.

`https://mybinder.org/v2/gh/jodivaso/examples-sage-kenzo/`
`master?filepath=examples.ipynb`

The interface loads Kenzo and disables the standard output. It defines new classes and methods which contain wrappers of the Kenzo ones. By means of Python commands (and calls to the `EclObject` and `ecl_eval` operators) it computes in Kenzo and provides the ouput as SageMath objects in a transparent way to the user.

The main class is `KenzoObject`, which inherits from `SageObject`, and is a wrapper to a Kenzo object (which is an ECL object). The other classes provide specific methods for each type of object (some of them are incorporated to connect the interface to Kenzo external modules). For instance, the `KenzoSimplicialSet` class possesses a method `homotopy_group`, which computes the n-th homotopy group of the simplicial set (via a call to the corresponding function in a Kenzo external package). The result of the method is an object of the `AbelianGroup` class, which is part of SageMath. Then, the user could invoke any operation provided by the `AbelianGroup` class over such an object.

It is worth distinguishing between two types of mathematical objects. On the one hand, there are objects (which are necessarily finite) that can be represented in both SageMath and Kenzo. Thus, they can be translated from SageMath to Kenzo and vice versa.

For some of these objects which are available in both Kenzo and SageMath, both systems use the same representation and then the translation between them is easy. This is the case of simplicial sets: an object of the SageMath class `SimplicialSet` can be translated to the wrapper class `KenzoSimplicialSet` defined in the interface via a conversion function called `KFiniteSimplicialSet`. Nevertheless, there are other structures such as chain complexes for which the representations in SageMath and Kenzo are different. As we have explained in Subsection 4.1.1.1, SageMath can construct a finite chain complex by means of either a dictionary with the differential matrices indexed by the elements of the grading group or, if the grading group is $\mathbb{Z}$, just a list of matrices. However, the Kenzo representation of a finite chain complex consists of a function that provides a list of generators for each degree and the differential maps described also as a function, as explained in Subsection 1.3.3.1. Thus, both representations share the same information (they represent the same finite chain complex), but in different format; then, we must provide a conversion function. When dealing with this case of different representations, our choice has been to implement the major part of the conversion in Lisp. This permits us to keep the interface as simple as possible and the connection is made via basic data types (integers, strings, lists, etc). This choice means that the development of the interface has required to code in both SageMath and Kenzo. In the case of finite chain complexes, we have implemented in Kenzo a function to construct a finite chain complex from a dictionary whose values are the differential matrices, i.e., following the same representation as SageMath uses. Then, the interface just presents `KChainComplex` as a wrapper to such a function.

Let us show the construction in SageMath of a chain complex from a dictionary $\{0 : m\}$, where m is the matrix $\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$.

```
sage: from sage.interfaces.kenzo import KChainComplex
sage: m = matrix(ZZ, 2, 2, [1, 0, 0, 0])
sage: sage_chcm = ChainComplex({0: m},
                                 degree_of_differential = -1)
sage: sage_chcm
Chain complex with at most 2 nonzero terms over Integer Ring
sage: type(sage_chcm)
<class 'sage.homology.chain_complex
                      .ChainComplex_class_with_category'>
```

Now, we compute the corresponding Kenzo object.

```
sage: kenzo_chcm = KChainComplex(sage_chcm)
```

```
sage: kenzo_chcm
[K37 Chain-Complex]
sage: type(kenzo_chcm)
<class 'sage.interfaces.kenzo.KenzoChainComplex'>
```

On the other hand, there are objects that are of infinite nature, and thus only Kenzo is able to represent them. In this case, the object that represents an infinite space is directly created in SageMath as a wrapper of the corresponding Kenzo object.

For instance, we have defined a function which returns the Eilenberg–MacLane space $K(\pi, n)$ as a Kenzo simplicial group, being π a cyclic group. To do that, we first define in the interface the corresponding ECL objects for $K(\mathbb{Z}, n)$, $K(\mathbb{Z}/2\mathbb{Z}, n)$ and $K(\mathbb{Z}/p\mathbb{Z}, n)$.

```
__k_z__ = EclObject("k-z")
__k_z2__ = EclObject("k-z2")
__k_zp__ = EclObject("k-zp")
```

Then, those functions are used in the implementation of the wrapper function, which returns an object which belongs to the class `KenzoSimplicialGroup`. That is, the implementation in SageMath internally calls the functions in Kenzo.

```
def EilenbergMacLaneSpace(G, n):
    if G == ZZ:
        kenzospace = __k_z__(n)
        return KenzoSimplicialGroup(kenzospace)
    elif G == AdditiveAbelianGroup([2]):
        kenzospace = __k_z2__(n)
        return KenzoSimplicialGroup(kenzospace)
    elif G in CommutativeAdditiveGroups() and G.is_cyclic():
        kenzospace = __k_zp__(G.cardinality(), n)
        return KenzoSimplicialGroup(kenzospace)
    else:
        raise NotImplementedError("...")
```

Let us note again that we have had to develop code for SageMath (the interface, the wrapper functions), but also for Kenzo. New functions have been added and some existing ones have been adapted in order to make the output compatible with the interface and SageMath. For example, we have created functions in Kenzo to create finite simplicial sets from the corresponding SageMath objects:

```
sage: from sage.interfaces.kenzo import KFiniteSimplicialSet
sage: CP3 = simplicial_sets.ComplexProjectiveSpace(3); CP3
CP^3
sage: KCP3 = KFiniteSimplicialSet(CP3); KCP3
[K39 Simplicial-Set]
```

Finally, let us present an example of execution in SageMath via Kenzo involving infinite structures. As we have already said, some direct computations of the interface of Kenzo in SageMath includes the construction of different algebraic and topological objects which are available in Kenzo such as spheres, Eilenberg–MacLane spaces, cartesian products, tensor products or loop spaces. Although some of them can also be constructed in SageMath, Kenzo makes it possible to work with elements of infinite nature so that it is interesting to include these functions in SageMath. For example, we can construct the cartesian product of the Eilenberg–MacLane space $K(\mathbb{Z}/2\mathbb{Z}, 2)$ and the loop space of the sphere S^3 and compute its homology groups as follows:

```
sage: from sage.interfaces.kenzo import EilenbergMacLaneSpace
sage: from sage.interfaces.kenzo import Sphere
sage: E = EilenbergMacLaneSpace(AdditiveAbelianGroup([2]), 2)
sage: S3 = Sphere(3)
sage: L = S3.loop_space()
sage: X = E.cartesian_product(L)
sage: [X.homology(i) for i in range(8)]
[Z, 0, Z x C2, 0, Z x C2 x C4, C2, Z x C2 x C2 x C4, C2 x C2]
```

Let us remark that the loop space of S^3 is not of finite type and then the space `X` is also of infinite nature. Moreover, the loop space constructor is not available in SageMath. Kenzo is able to compute the homology groups of our space `X` using the effective homology technique explained in Subsection 1.3.2.

4.2.2 Participation in Google Summer of Code 2020

Google Summer of Code (GSoC) is a global program focused on introducing students to open source software development. Students work on a 10 week programming project with an open source organization and they are paired with a mentor from the participating organizations.

In the GSoC 2020, SageMath was selected as one of the open source organizations and we participated in the project *Integration of Kenzo program with `SimplicialSets`*. Our proposal consisted in improving the interface between the Kenzo Program and SageMath, whose initial version was worked in ticket `https://trac.sagemath.org/ticket/27880` (described in Subsection 4.2.1). Specifically, our purpose was to combine the functions and techniques of algebraic topology for simplicial sets developed in Kenzo and those existing in the module `SimplicialSets` of SageMath, in order to complement and enrich the functionalities for computing invariants of simplicial sets.

We created the ticket

`https://trac.sagemath.org/ticket/29879`

for the development of the project.

The following items show some descriptions and examples of the changes made on the Kenzo and SageMath files:

`kenzo.py`

This is the file where the wrapping functions are defined and implemented for Kenzo objects. The initial version of this file corresponds to the interface presented in the previous subsection. The most important changes to this file are summarized in the next lines.

- The building blocks for simplicial sets are abstract simplices. These objects are implemented in Kenzo as a special type `ABSM`. We considered important to create a class `KenzoAbstractSimplex` as a wrapper for this type of Kenzo objects. Different methods were implemented for this class: the non-degenerate part, the list of degeneracy operators applied to it (if they exist) and the cartesian product with other abstract simplices.
- In Kenzo, the non-degenerate simplices of a cartesian product of simplicial sets are represented internally by an object of type `CRPR`. In order to wrap this objects, the class `KenzoCRPRSimplex` was created and a method to return the factors that generate the abstract simplex in the cartesian product was implemented.
- Combinations in Kenzo are objects of type `CMBN` representing a sum of terms in a chain complex. The class `KenzoCombination` was implemented to wrap these objects and methods wrapping Kenzo functions dealing with combinations were also created: the opposite of a combination, the sum, substraction and scalar multiplication. The function `Kenzocmbn` was implemented in order to create an instance of `KenzoCombination` from an ECL object representing a combination.
- In the class `KenzoChainComplexMorphism` we have added methods for the opposite of a morphism, the sum, substraction, composition and scalar multiplication of morphisms.
- Also, we have created the class `KenzoSimplicialSetMorphism` to wrap simplicial set morphisms in Kenzo. Methods for computing the cone, the suspension and the pushout of simplicial set morphisms have been created to deal with these objects. The function `KSimplicialSetMorphism` was implemented to create a `KenzoSimplicialSetMorphism` from an instance of `SimplicialSetMorphism` in SageMath.
- The functions `Kenzo_from_IdNumber` and `IdNumber` were implemented in order to use the slot `:idnm` in some Kenzo objects (like simplicial sets, chain complexes, morphisms, spectral sequences...) to identify and recover the respective object in Kenzo. Also, the method `orgn` was implemented in classes were the Kenzo objects have `:orgn` slot.

`sage-interface.lisp`

In this Common Lisp file, the Kenzo functions are implemented in order to be imported in `kenzo.py`.

- In order to provide the slot `:intr-dffr` of the `KenzoSimplicialSet` in SageMath obtained by applying the function `KChainComplex` (in `kenzo.py`) to a `ChainComplex` in SageMath, the function `KDFFR` was implemented.
- The function `KINTR` was added to provide the slot `:intr` of the instance of `KenzoChainComplexMorphism` in SageMath that is obtained by applying the function `KMorphismChainComplex` (in `kenzo.py`) to an instance object of the class `ChainComplexMorphism` in SageMath. In a similar way, the function `KSINTR` was added to provide the slot `:sintr` of the instance of `KenzoSimplicialMorphism` in SageMath obtained by applying the function `KSimplicialSetMorphism` (in `kenzo.py`) to an instance object of `SimplicialSetMorphism` in SageMath.

All the functions and methods created in `kenzo.py` and `sage-interface.lisp`, provide a better communication between SageMath and Kenzo. In addition, we have added the attribute `_kenzo_repr` to a wide variety of classes and constructors of simplicial sets and chain complexes developed by other authors in SageMath. This attribute permits to assign (whenever it is possible) to each object `S` in such classes, an equivalent object in the Kenzo system (an ECL object `S._kenzo_repr`), which is automatically constructed when `S` is created, allowing to integrate both systems in a better way and to use Kenzo functionalities directly as methods of Sage objects. In order to build these objects, we use the functions defined in the file `kenzo.py`.

Let us describe and show some examples of computations involving `_kenzo_repr`.

`chain_complex.py`

The `_kenzo_repr` attribute was added to `ChainComplex_class`, developed by J. Palmieri: when a `ChainComplex` in SageMath has $\mathbb{Z}$ as grading group and the degree of differentials is -1, its Kenzo representation is constructed (these are restrictions given by the Kenzo system). For example,

```
sage: m1 = matrix(ZZ, 3, 2, [-1, 1, 3, -4, 5, 6])
sage: m4 = matrix(ZZ, 2, 2, [1, 2, 3, 6])
sage: m5 = matrix(ZZ, 2, 3, [2, 2, 2, -1, -1, -1])
sage: C = ChainComplex({1: m1, 4: m4, 5: m5},
                           degree_of_differential = -1)
sage: KC = C._kenzo_repr; KC
[K1 Chain-Complex]
sage: type(KC)
<class 'sage.interfaces.kenzo.KenzoChainComplex'>
```

In case the differential maps of a `ChainComplex` do not have degree -1, the attribute `_kenzo_repr` is not assigned:

```
sage: m = matrix(ZZ, 2, 3, [3, 0, 0, 0, 0, 0])
sage: D = ChainComplex({0: m})
sage: hasattr(D, '_kenzo_repr')
False
sage: hasattr(D.dual(), '_kenzo_repr')
True
```

`chain_complex_morphism.py`

The `_kenzo_repr` attribute was added to the class `ChainComplexMorphism`: if the source complex and the target complex have `_kenzo_repr` attributes, the Kenzo representation of the morphism is constructed.

Different operations between `ChainComplexMorphisms` have Kenzo representations: the opposite of a morphism, sum of morphisms, composition of morphisms, multiplication by an integer number, substraction of morphisms. All of these operations have `_kenzo_repr` attributes whenever the involved morphisms have Kenzo representations.

Observe that `simplicial_complexes.Sphere(2)` has a `_kenzo_repr` attribute. If we define the following `ChainComplexMorphism`:

```
sage: S = simplicial_complexes.Sphere(2)
sage: H = Hom(S,S)
sage: i = H.identity()
sage: x = i.associated_chain_complex_morphism()
sage: w = -x
sage: z = x + x
```

we can call their corresponding Kenzo representations:

```
sage: Kx = x._kenzo_repr; Kx
[K7 Morphism (degree 0): K5 -> K5]
sage: Kw = w._kenzo_repr; Kw
[K9 Morphism (degree 0): K5 -> K5]
sage: Kz = z._kenzo_repr; Kz
[K11 Morphism (degree 0): K5 -> K5]
```

Since `w` is the opposite of `x` in SageMath, it implies that `Kw` is the opposite morphism of `Kx` in Kenzo:

```
sage: Kw.orgn()
'(OPPS [K7 Morphism (degree 0): K5 -> K5])'
```

In the same way, note that in SageMath, `z` is the sum of `x` with itself. Then, `Kz` is automatically interpreted in Kenzo as the sum of the morphism `Kx` with itself:

```
sage: Kz.orgn()
'(2MRPH-ADD [K7 Morphism (degree 0): K5 -> K5]
            [K7 Morphism (degree 0): K5 -> K5]
            :CMBN)'
```

simplicial_set.py

The _kenzo_repr attribute was added to AbstractSimplex_class (for this, the implementation of the class KenzoAbstractSimplex in kenzo.py was necessary). Different methods of this class have their respective Kenzo representations, for example, the list of degeneracies and the nondegenerate part of an abstract simplex:

```
sage: from sage.homology.simplicial_set \
      import AbstractSimplex
sage: v = AbstractSimplex(3, (0,1,3), name = 'v')
sage: Kv = v._kenzo_repr; Kv
<AbSm 5-2-0 S8763998853024>
sage: Kv.degeneracies()
[5, 2, 0]
sage: Kv.degeneracies() == v.degeneracies()
True
```

The class KenzoCRPRSimplex in kenzo.py was created in order to represent the product of abstract simplices:

```
sage: w = AbstractSimplex(2, (0,1), name = 'w')
sage: Kw = w._kenzo_repr; Kw
<AbSm 2-0 S8763998853087>
sage: pr = Kv.product(Kw); pr
<AbSm 2-0 <CrPr 3 S8763998853024 - S8763998853087>>
sage: pr.nondegenerate()
<CrPr 3 S8763998853024 - S8763998853087>
sage: pr.nondegenerate().factors()
(<AbSm 3 S8763998853024>, <AbSm - S8763998853087>)
```

In the class SimplicialSet_arbitrary, the join method is not implemented, but in Kenzo this operation is available. The join of simplicial sets is created as a KenzoSimplicialSet, whenever the involved simplicial sets have _kenzo_repr attributes. In the following example, the join S2.join(S2) is computed by means of the instruction (join S2._kenzo_repr S2._kenzo_repr) in Kenzo:

```
sage: S2 = simplicial_sets.Sphere(2)
sage: M = S2.join(S2); M
[K38 Simplicial-Set]
sage: [M.homology(i) for i in range(7)]
[Z, 0, Z, Z, 0, Z, 0]
```

In the class `SimplicialSet_finite`, the `_kenzo_repr` attribute was also added.

```
sage: from sage.homology.simplicial_set \
      import AbstractSimplex, SimplicialSet
sage: v = AbstractSimplex(0)
sage: e = AbstractSimplex(1)
sage: S = SimplicialSet({e: (v, v)})
sage: KS = S._kenzo_repr; KS
[K228 Simplicial-Set]
sage: KS.orgn()
'(BUILD-FINITE-SS (CELL_0_0 1 CELL_1_0
                   ((CELL_0_0) (CELL_0_0))))'
```

`simplicial_set_constructions.py`

In class `PullbackOfSimplicialSets_finite`, the stored data in the attribute `self._translation` is used to “translate” the involved simplices to their Kenzo equivalent abstract simplices. Since the product of simplicial sets is implemented in SageMath as a pullback, this translation allows us to associate the `_kenzo_repr` attribute to the class `ProductOfSimplicialSets`:

```
sage: S1 = simplicial_sets.Sphere(1)
sage: KS1 = S1._kenzo_repr
sage: T = simplicial_sets.Torus()
sage: KT = T._kenzo_repr
sage: KT.orgn()
'(CRTS-PRDC [K233 Simplicial-Set] [K233 Simplicial-Set])'
sage: KT._factors[0] == KT._factors[1] == KS1
True
sage: KT == KS1.cartesian_product(KS1)
True
```

In class `PushoutOfSimplicialSets`, the `_kenzo_repr` attribute was added bearing in mind that (by now) only pushouts of two morphisms are constructed in Kenzo.

```
sage: K = simplicial_sets.Simplex(4)
sage: L = K.n_skeleton(3)
sage: S4 = L.pushout(L.constant_map(), L.inclusion_map())
sage: KS4 = S4._kenzo_repr
sage: KS4.orgn()
'(PUSHOUT [K271 Simplicial-Morphism K265 -> K238]
          [K270 Simplicial-Morphism K265 -> K260])'
sage: Kf0 = S4._maps[0]._kenzo_repr
sage: Kf1 = S4._maps[1]._kenzo_repr
sage: Kf0.pushout(Kf1) == KS4
```

```
True
```

The _kenzo_repr attribute was also added to WedgeOfSimplicialSets and SmashProductOfSimplicialSets, where the Kenzo analog is constructed if all the involved factors have their Kenzo representations.

```
sage: T = simplicial_sets.Torus()
sage: KT = T._kenzo_repr
sage: S2 = simplicial_sets.Sphere(2)
sage: KS2 = S2._kenzo_repr
sage: P = T.smash_product(S2)
sage: KP = P._kenzo_repr
sage: KP.orgn()
'(PUSHOUT [K344 Simplicial-Morphism K339 -> K298]
          [K345 Simplicial-Morphism K339 -> K327])'
sage: P.homology()
{0: 0, 1: 0, 2: 0, 3: Z x Z, 4: Z}
sage: [KP.homology(i) for i in range(5)]
[Z, 0, 0, Z x Z, Z]
```

If a simplicial set has_kenzo_repr attribute, its suspension has a Kenzo representation (implemented in class SuspensionOfSimplicialSet).

```
sage: S3 = simplicial_sets.Sphere(3)
sage: S4 = S3.suspension()
sage: KS3 = S3._kenzo_repr
sage: KS4 = S4._kenzo_repr
sage: KS4.orgn()
'(SUSPENSION [K748 Simplicial-Set])'
sage: KS4._base == KS3
True
```

simplicial_set_examples.py

In the function Sphere was assigned the _kenzo_repr attribute in order to construct a Kenzo sphere, which is possible when the dimension of the sphere is less than 15 (the maximal dimension allowed in Kenzo system).

```
sage: S6 = simplicial_sets.Sphere(6)
sage: KS6 = S6._kenzo_repr
sage: KS6.orgn()
'(SPHERE 6)'
sage: [KS6.homology(i) for i in range(7)]
[Z, 0, 0, 0, 0, 0, Z]
```

Also, in Simplex function the Kenzo analog (DELTA function) was added as the _kenzo_repr attribute.

```
sage: D2 = simplicial_sets.Simplex(2)
sage: KD2 = D2._kenzo_repr
sage: KD2.orgn()
'(DELTA 2)'
sage: (KD2.basis(0), KD2.basis(1), KD2.basis(2))
([1, 2, 4], [3, 5, 6], [7])
```

In the function `RealProjectiveSpace` the `_kenzo_repr` attribute was assigned allowing the construction of the analog finite or infinite dimensional real projective space in Kenzo.

```
sage: RP5 = simplicial_sets.RealProjectiveSpace(5)
sage: KRP5 = RP5._kenzo_repr
sage: KRP5.orgn()
'(R-PROJ-SPACE 1 6)'
sage: [KRP5.homology(i) for i in range(6)]
[Z, C2, 0, C2, 0, Z]
sage: BC2 = simplicial_sets.RealProjectiveSpace(Infinity)
sage: KBC2 = BC2._kenzo_repr
sage: KBC2.orgn()
'(R-PROJ-SPACE 1 :INFINITY)'
```

`simplicial_set_morphism.py`

In the class `SimplicialSetMorphism`, the `_kenzo_repr` attribute is assigned when the domain and the codomain simplicial sets have Kenzo representations. Since the definition of a simplicial set morphism in SageMath is stored in the attribute `_dictionary`, we have used this data to construct the Kenzo representation of this simplicial set morphisms.

```
sage: S5 = simplicial_sets.Sphere(5)
sage: s = S5.n_cells(5)[0]
sage: one = S5.Hom(S5)({s: s}); one
Simplicial set endomorphism of S^5
  Defn: Identity map
sage: Kone = one._kenzo_repr; Kone
[K844 Simplicial-Morphism K839 -> K839]
sage: type(Kone)
<class 'sage.interfaces.kenzo.KenzoSimplicialSetMorphism'>
```

Some constructions involving simplicial set morphisms like cones, suspensions and pushouts, which are implemented in SageMath, have their respective Kenzo representations.

```
sage: T = simplicial_sets.Torus()
sage: K = simplicial_sets.KleinBottle()
sage: init_T = T._map_from_empty_set()
```

```
sage: init_K = K._map_from_empty_set()
sage: D = init_T.pushout(init_K) # the disjoint union
                                 # as a pushout
sage: Kinit_T = init_T._kenzo_repr
sage: Kinit_K = init_K._kenzo_repr
sage: KD = D._kenzo_repr
sage: Kinit_T.pushout(Kinit_K) == KD
True
```

4.3 Including finite topological spaces in SageMath

There exist some external modules to SageMath implementing functions to deal with finite topological spaces [Fer17a], [Ren19]. However, these implementations have not been included in tickets for the evaluation of SageMath developers and therefore there is no integration with other functionalities of SageMath. We have created a module implementing finite topological spaces and related concepts by using our previously explained Kenzo algorithms as part of the tickets

https://trac.sagemath.org/ticket/30400
https://trac.sagemath.org/ticket/30447
https://trac.sagemath.org/ticket/30862

Let us describe our new module in SageMath. The class `FiniteTopologicalSpace` has been created by taking as input the following parameters (attributes for the instances of the classes):

`elements` is the list of elements of the finite space.

`minimal_basis` is a dictionary where the keys are the points `n` in `elements` and `minimal_basis[n]` is the set of points in the minimal open set containing `n`.

`topogenous` is the topogenous matrix $T = [t_{ij}]$ of the finite space induced by the order given in `elements`, that is, $t_{ij} = 1$ if `elements[i-1]` $\leqslant$ `elements[j-1]` and $t_{ij} = 0$ otherwise.

Let us note that the cardinality of the finite space is computed and saved as an attribute `self._cardinality`. Since finite T_0-spaces are very important in the study of invariants of finite topological space, we have created the subclass `FiniteTopologicalSpace_T0`, inheriting all the attributes above mentioned (the `self._topogenous` is assumed to be upper triangular) and, additionally, the parameter

`poset` (default `None`) is a `FinitePoset` representing the partially ordered set corresponding to the finite T_0-space (by Alexandroff correspondence).

The function `FiniteSpace` has been implemented to create a finite space by using the following parameters:

`data` takes any of the following forms:

1. A dictionary representing the minimal basis of the space. The keys are taken as the elements `x` of the space, and the respective `data[x]` must be the minimal open set of `x`.
2. A list or tuple of minimal open sets (in this case the elements of the space are assumed to be `range(n)`, that is, the integers $0, \dots, \mathtt{n}-1$, where `n` is the length of `data`).
3. A topogenous matrix (assumed sparse). If `elements = None`, the elements of the space are assumed to be `range(n)` where `n` is the dimension of the matrix.
4. A `FinitePoset`.

`elements` parameter (default `None`) is ignored when `data` is of type 1, 2 or 4. In case `data` is a topogenous matrix, `elements` must be a list, a tuple or a set representing the underlying set of the finite space.

`is_T0` (default `False`) is a boolean that indicates, when it is known, if the finite space satisfies the T_0 separation axiom.

For example, by specifying `data` as a dictionary we can construct finite spaces:

```
sage: FiniteSpace({'a': {'a', 'c'},
                   'b': {'b'},
                   'c': {'a', 'c'}})
Finite topological space of 3 points with minimal basis
{'a': {'a', 'c'}, 'b': {'b'}, 'c': {'a', 'c'}}
```

Observe that, even when the parameter `is_T0` is not fixed as `True`, the T_0 axiom is checked in `data`:

```
sage: FiniteSpace([{0, 3}, {1, 3}, {2, 3}, {3}])
Finite T0 topological space of 4 points with minimal basis
{0: {3, 0}, 1: {3, 1}, 2: {3, 2}, 3: {3}}
```

The system checks that the corresponding `data` define a topology on the underlying set:

```
sage: FiniteSpace({'a': {'a', 'b'}})
Traceback (most recent call last):
...
ValueError: The data does not correspond to a valid dictionary
sage: FiniteSpace({'a': {'a', 'b'},
                   'b': {'a', 'b'},
                   'c': {'a', 'c'}})
```

```
Traceback (most recent call last):
...
ValueError: The introduced data does not define a topology
```

We have implemented functions to compute some topological properties. For instance, the module provides functions to determine if a point is interior, exterior, boundary, limit, closure or isolated with respect to a given subspace. Moreover, for a given subspace, the corresponding interior, exterior, boundary, derived, closure or isolated set can be computed.

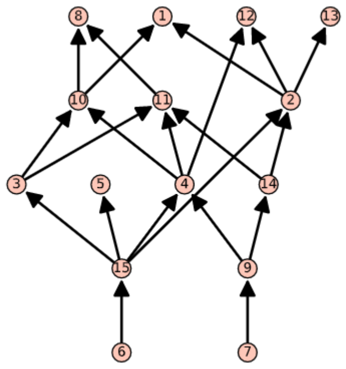
```
sage: import random
sage: T = FiniteSpace(posets.RandomPoset(30, 0.5))
sage: X = T.underlying_set()
sage: k = randint(0,len(X))
sage: E = set(random.sample(X, k))
sage: Fr = T.boundary(E)
sage: Cl = T.closure(E)
sage: Der = T.derived(E)
sage: Int = T.interior(E)

sage: Fr == T.closure(E) - T.interior(E)
True
sage: X == Fr.union(T.interior(E), T.exterior(E))
True
sage: T.interior(T.boundary(Cl)) == set()
True
sage: T.closure(E) == E.union(Der)
True
sage: Int == X - T.closure(X - E)
True
```

In case `data` define a finite T_0-space (for example, when `data` is a poset), we can use the methods and functions implemented in the class `FinitePoset`. In particular, it is very useful to visualize the Hasse diagrams of the posets associated to finite T_0-spaces by using graphic objects in SageMath:

```
sage: minimal_basis = {7: {7}, 9: {9, 7}, 14: {9, 14, 7},
                       6: {6}, 15: {6, 15}, 5: {5, 6, 15},
                       4: {4, 6, 7, 9, 15}, 3: {3, 6, 15},
                       10: {3, 4, 6, 7, 9, 10, 15},
                       11: {3, 4, 6, 7, 9, 11, 14, 15},
                       8: {3, 4, 6, 7, 8, 9, 10, 11, 14, 15},
                       2: {2, 6, 7, 9, 14, 15},
                       12: {2, 4, 6, 7, 9, 12, 14, 15},
                       13: {2, 6, 7, 9, 13, 14, 15},
                       1: {1, 2, 3, 4, 6, 7, 9, 10, 14, 15}}
```

```
sage: X = FiniteSpace(minimal_basis); X
Finite T0 topological space of 15 points
sage: X.show()
```

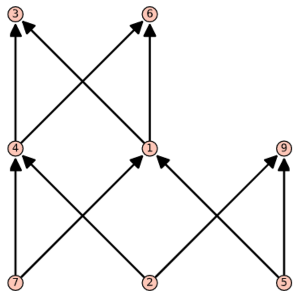


For a finite space X not satisfying the T_0 axiom, we can construct the space X_0 in Proposition 1.28 by using the method `equivalent_T0`.

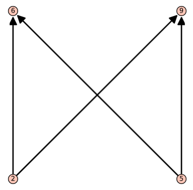
```
sage: minimal_basis = ({0}, {1, 3, 4}, {0, 2, 5}, {1, 3, 4},
                       {1, 3, 4}, {0, 2, 5}, {0, 6, 7},
                       {0, 6, 7}, {0, 2, 5, 8}, {9})
sage: X = FiniteSpace(minimal_basis); X
Finite topological space of 10 points
sage: X.is_T0()
False
sage: X.equivalent_T0()
Finite T0 topological space of 6 points with minimal basis
 {0: {0}, 1: {1}, 2: {0, 2}, 6: {0, 6}, 8: {0, 2, 8}, 9: {9}}
```

By using the above construction, the implemented methods to compute invariants on finite T_0-spaces can be applied to find invariants of arbitrary finite topological spaces. In particular, identification of beat points, weak points and construction of cores and weak cores can be performed by using our module:

```
sage: minimal_basis = {7: {7}, 5: {5}, 8: {5, 7, 8}, 2: {2},
                       4: {2, 4, 7}, 9: {2, 5, 9},
                       1: {1, 5, 7, 8},
                       3: {1, 2, 3, 4, 5, 7, 8},
                       6: {1, 2, 4, 5, 6, 7, 8}}
sage: X = FiniteSpace(minimal_basis)
sage: X.core().show()
```

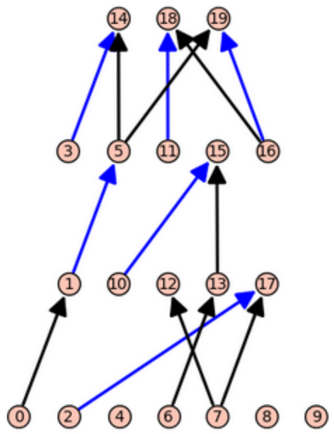


```
sage: X.core([1, 2, 3, 4, 5, 6, 7, 8])
Finite T0 topological space of 1 points with minimal basis
 {8: {8}}
sage: X.weak_core().show()
```



Finally, the interface between Kenzo and SageMath allows us to use the algorithm to compute discrete vector fields and homology of h-regular spaces described in Chapter 2.

```
sage: X = RandomFiniteT0Space(20, 0.1)
sage: dvf = X.discrete_vector_field(); dvf
[(1, 5), (3, 14), (10, 15), (2, 17), (11, 18), (16, 19)]
sage: X.show(highlighted_edges = dvf)
```



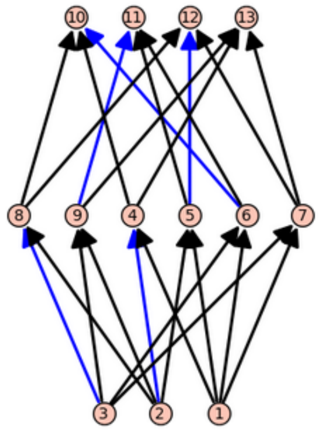
```
sage: covers = [[9, 13], [7, 13], [4, 13], [8, 12],
                [7, 12], [5, 12], [9, 11], [6, 11],
                [5, 11], [8, 10], [6, 10], [4, 10],
                [3, 9], [2, 9], [3, 8], [2, 8],
                [3, 7], [1, 7], [3, 6], [1, 6],
                [2, 5], [1, 5], [2, 4], [1, 4]]
sage: P = Poset((list(range(1,14)), covers),
                cover_relations = True)
sage: X = FiniteSpace(P)
```

We compute the homology by using the general method:

```
sage: X.hregular_homology()
{0: Z, 1: C2, 2: 0}
```

Now, we determine homology by applying discrete vector fields and the corresponding h-regular homology algorithm:

```
sage: dvf = X.discrete_vector_field(); dvf
[(3, 8), (2, 4), (6, 10), (9, 11), (5, 12)]
sage: X.hregular_homology(dvfield = dvf)
{0: Z, 1: C2, 2: 0}
sage: X.show(highlighted_edges = dvf)
```



Chapter 5

Strategies to compute discrete vector fields

One of the main goals of Discrete Morse Theory is to find *long* discrete vector fields on a topological space in order to have a *small* number of critical points, since these data are enough to describe the homotopy type of such space [For98]. Moreover, Discrete Morse Theory has been used to develop combinatorial methods to study n-deformations of regular CW-complexes and applications to the study of the Andrews-Curtis conjecture [Fer19], by using the face posets of the CW-complexes. Regarding to the computation of discrete vector fields defined on a finite topological space, the result [CO17, Theorem 4.3] presented in Section 2.3 establishes that homology groups of the initial chain complex associated with the finite T_0-space are isomorphic to those of a smaller chain complex generated by the critical cells defined from the vector field.

Therefore, it is natural to consider suitable strategies to compute *better* homologically admissible Morse matchings on the Hasse diagrams associated to finite T_0-spaces, than those we have computed by using the algorithm described in Subsection 2.3.1.

In this chapter we present some partial results about different strategies we have considered to compute discrete vector fields on finite T_0-spaces. First, we describe some natural strategies to choose a new vector to be added to a vector field, that is, its column and row indexes on the Stong matrix. These strategies can be used to enhance the algorithm presented in Subsection 2.3.1. Moreover, we consider some machine learning techniques such as reinforcement learning and Monte-Carlo tree search to obtain discrete vector fields as big as possible. The code used in this chapter is available at [CR20] in the folder /dvfields-strategies.

5.1 Strategies description

The algorithm we have developed to compute discrete vector fields on finite T_0-spaces (see Subsection 2.3.1) searches for homologically admissible edges such that the set of such edges does not contains cycles in the modified Hasse diagram of the poset associated to the finite space (remember that (x, y) is homologically admissible if $\widehat{U}_y \setminus \{x\}$ is homotopically trivial). In this searching, we go through the columns and the rows of the Stong matrix. This walk can be made in ascending order, as was implemented in our function `dvfield` presented in Subsection 2.3.1, or by other ways to sort the columns and rows indexes of a matrix.

We are interested in comparing different *strategies*, by analyzing if there exist remarkable differences between the size of the discrete vector fields obtained in each case. Remember that the longer the discrete vector field is, the smaller the number of generators we need to describe a chain complex to compute homology by means of the critical complex is.

In order to illustrate some possible strategies to compute discrete vector fields on finite T_0-spaces, let us consider the finite space in Figure 5.1.

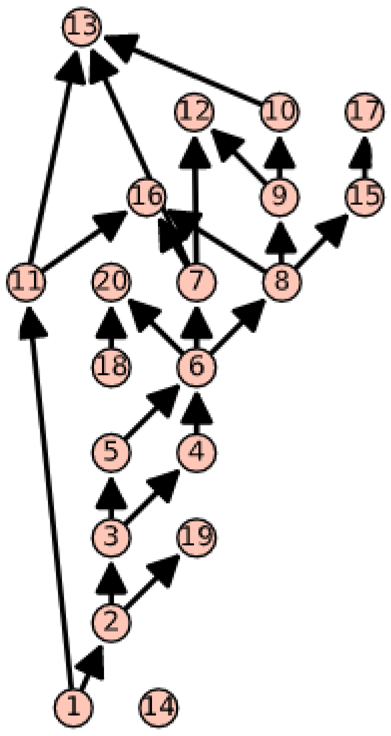


Figure 5.1: Hasse diagram of the finite T_0-space `X`.

```
> (setf edges '((1 2) (2 3) (3 4) (3 5) (4 6) (5 6) (6 7)
                (6 8) (8 9) (9 10) (1 11) (7 12) (9 12)
                (7 13) (10 13) (11 13) (8 15) (7 16) (8 16)
                (11 16) (15 17) (2 19) (6 20) (18 20)))
> (setf X (build-finite-space
           :stong (edges-to-stong 20 edges)
           :orgn '(X)))
```

The order of the elements of `X` is given as usual in Kenzo, that is, in ascending order. We have considered six different strategies to sort a list of elements in a finite T_0-space: *:standard*, *:random*, *:indegree*, *:reverse-indegree*, *:outdegree* and *:reverse-outdegree*. Taking into account the next function:

`dvfield-strategies-col-row` *finspace strategy-cols strategy-rows*
: It returns a discrete vector field on the space *finspace*. The searching of the vectors is carried out by sorting the columns (resp. rows) of the Stong matrix by using the strategy given by *strategy-cols* (resp. *startegy-rows*).

Let us describe the strategies in the next lines:

:standard. This strategy refers to sort a set of elements in ascending order (remember that in Kenzo, the elements of a finite T_0-space are integer numbers starting by 1). This is the strategy used to sort column and row indexes in the function `dvfield` in Chapter 2. In fact, this is the ordering proposed in the computation of discrete vector fields on chain complexes in Kenzo (see Figure 1.10). By using this sorting, we obtain the next discrete vector field:

```
> (dvfield-strategies-col-row X :standard
                                 :standard)
((2 3) (4 6) (8 9) (7 12) (10 13)
 (11 16) (15 17) (18 20))
```

Note that the edge `(1 2)` was the first candidate to be a vector, but the subspace $\widehat{U}_2 \setminus \{1\} = \emptyset$ is not homotopically trivial, then it was not chosen.

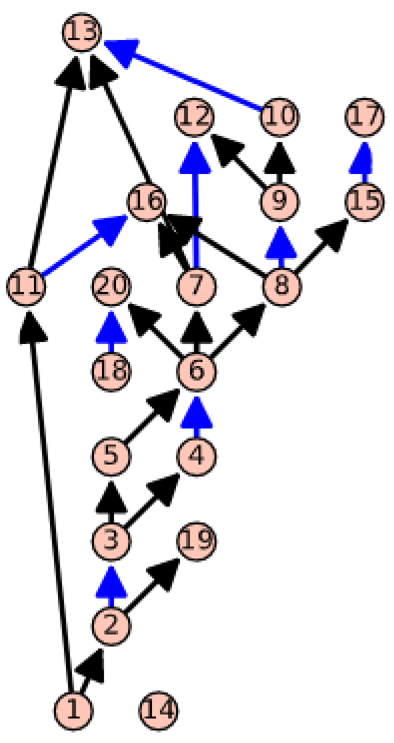
:random. Random orderings of rows and columns can be considered, meaning that there is no preferable choice of the indexes in the searching we are doing. A discrete vector field that we obtained by using this strategy is the following:

```
> (dvfield-strategies-col-row X :random
                                 :random)
((2 19) (18 20) (7 16) (3 5) (8 15)
 (9 12) (4 6) (11 13))
```

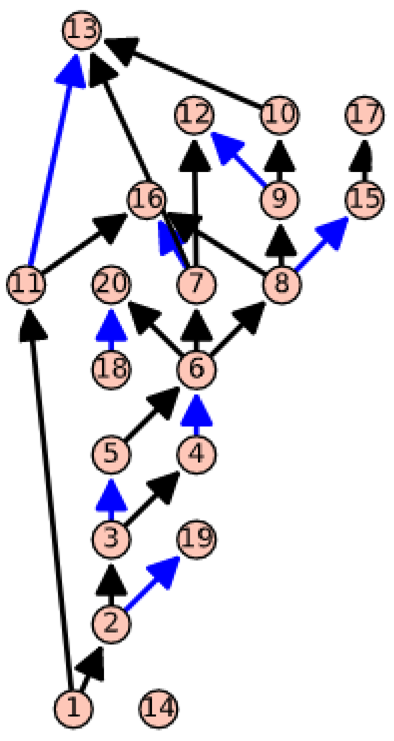
:indegree and *:reverse-indegree*. The ***indegree*** of a vertex v in a directed graph, denoted by $\deg^{-}(v)$, is the number of "head ends" adjacent to it. Note that it seems natural to prefer a vertex with low indegree as a suitable candidate for being part of a vector in a discrete vector field, since in future steps all the edges which have such vertex as head can not be candidates to vector.

We can use this criteria to sort the elements of a list. In *:indegree* strategy, we sort the elements of a set by the rule $x \preccurlyeq y$ if $\deg^{-}(x) \leqslant \deg^{-}(y)$. By using this rule, the elements of `X` are sorted as follows:

```
> (sorting-by-strategy X :indegree)
(1 14 18 2 3 4 5 7 8 9 10 11 15 17 19 6 12 20 13 16)
```



:standard



:random

Figure 5.2: Discrete vector fields on `X` (in blue) by using *:standard* and *:random* strategies.

In order to compute a discrete vector field by using this strategy, observe that the first three elements `1`, `14` and `18` have indegree index equal to zero (see Figure 5.1), so that they are no tail of any edge and then, in their respective columns in the Stong matrix of `X`, there are no ones (different of the diagonal entries). The next element to be checked is `2`; then, the edge `(1 2)` is the first candidate to be a vector, but the subspace $\widehat{U}_2 \setminus \{1\} = \emptyset$, is not homotopically trivial.

In the column of the Stong matrix corresponding to the next element `3`, the edge `(2 3)` is identified as the first vector to be added to the vector field because such edge satisfies all the conditions to be included in it. The next two elements, `4` and `5`, have only one edge coming from `3`, then the edges `(3 4)` and `(3 5)` can not be part of the vector field (since the element `3` is already a target). The next element `7` is the head of the edge `(6 7)`, which is included in the vector field since it satisfies all the conditions. Continuing with this process, we obtain a discrete vector field:

```
> (dvfield-strategies-col-row X :indegree
                                :indegree)
((2 3) (6 7) (8 9) (15 17)
 (18 20) (10 13) (11 16))
```

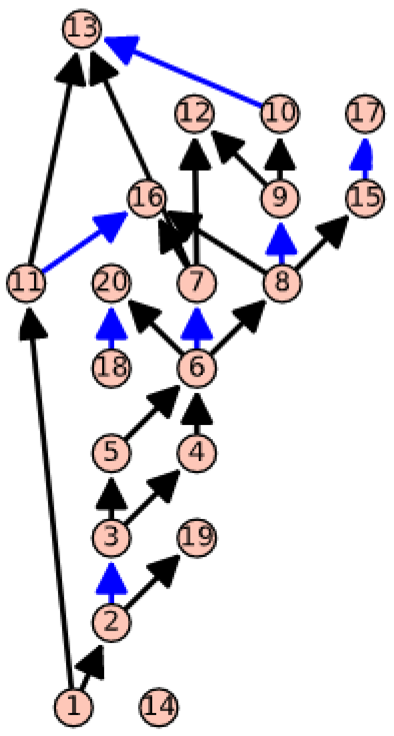
The *:reverse-indegree* strategy is considered by using the reverse order:

```
> (sorting-by-strategy X :reverse-indegree)
(13 16 6 12 20 2 3 4 5 7 8 9 10 11 15 17 19 1 14 18)
```

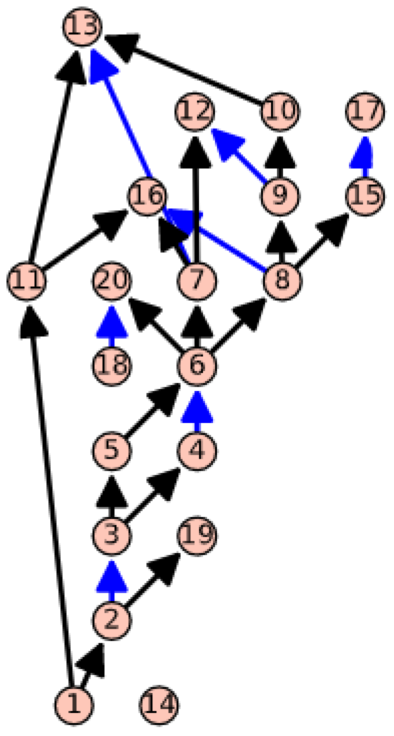
And the corresponding discrete vector field is given by:

```
> (dvfield-strategies-col-row X :reverse-indegree
                                :reverse-indegree)
((7 13) (8 16) (4 6) (9 12) (18 20) (2 3) (15 17))
```

The discrete vector fields found with the two above strategies applied on X are illustrated in Figure 5.3.



:indegree



:reverse-indegree

Figure 5.3: Discrete vector fields on X (in blue) by using *:indegree* and *:reverse-indegree* strategies.

:outdegree and *:reverse-outdegree*. The ***outdegree*** of a vertex v in a directed graph, denoted by $\deg^+(v)$, is the number of "tail ends" adjacent to it. Observe that once a vertex v is chosen to be part of a vector in a discrete vector field, all the edges having v as tail can not be candidates to vector in future steps, so that it seems preferable to choose vertices with low outdegree.

Then, in *:outdegree* strategy, we sort the elements of a set by the rule $x \preccurlyeq y$ if $\deg^+(x) \leqslant \deg^+(y)$. By using this criteria, the elements of `X` can be sorted as follows:

```
> (sorting-by-strategy X :outdegree)
(12 13 14 16 17 19 20 4 5 10 15 18 1 2 3 9 11 6 7 8)
```

By using this ordering, the corresponding discrete vector field is obtained:

```
> (dvfield-strategies-col-row X :outdegree
                                  :outdegree)
((9 12) (10 13) (11 16) (15 17)
 (2 19) (18 20) (3 4) (5 6))
```

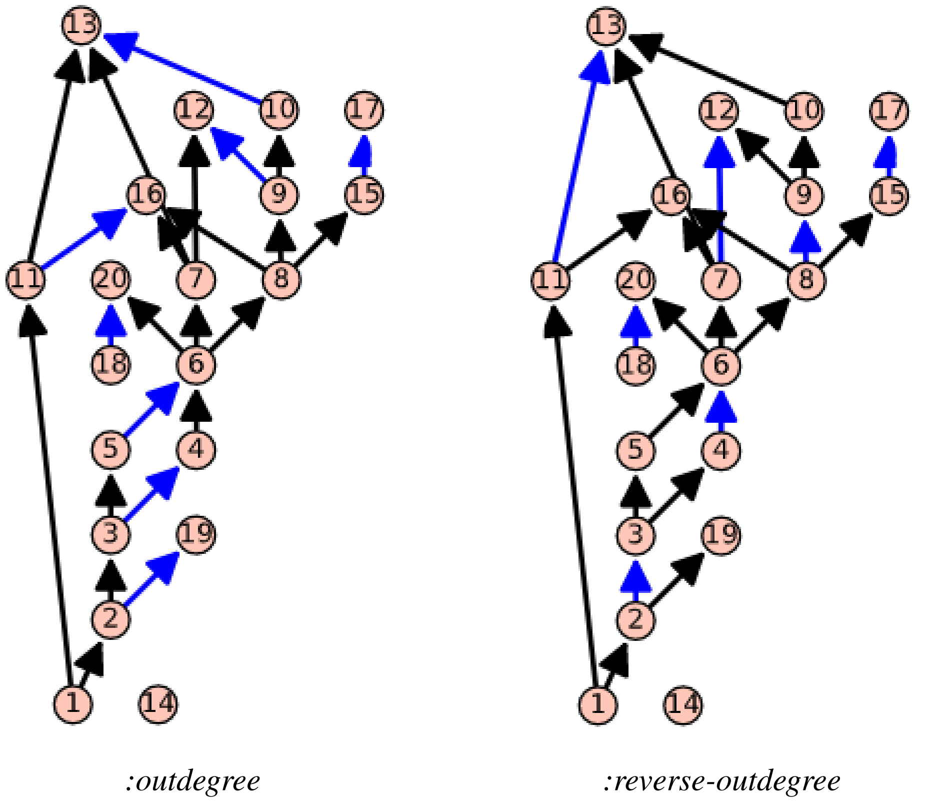


Figure 5.4: Discrete vector fields on `X` (in blue) by using *:outdegree* and *:reverse-outdegree* strategies.

The reverse order is used in the *:reverse-outdegree* strategy:

```
> (sorting-by-strategy X :reverse-outdegree)
(6 7 8 1 2 3 9 11 4 5 10 15 18 12 13 14 16 17 19 20)
```

The discrete vector field obtained by using the above ordering of rows and columns in the Stong matrix of `X`, is given by:

```
        > (dvfield-strategies-col-row X :reverse-outdegree
                                        :reverse-outdegree)
        ((4 6) (2 3) (8 9) (7 12) (11 13) (15 17) (18 20))
```

In Figure 5.4, the last two discrete vector fields are illustrated on X.

Observe that in the finite T_0-space X, the length of the discrete vector fields obtained by using *:standard*, *:random* and *:outdegree* is equal to 8, while those computed by using *:indegree*, *:reverse-indegree* and *:reverse-outdegree* are of length 7. These discrete vector fields were obtained by sorting columns and rows indexes of the Stong matrix of X with the same strategy, but if we consider *:outdegree* for columns and *:reverse-outdegree* for rows, we obtain the next discrete vector field of length 9 (see Figure 5.5):

```
> (dvfield-strategies-col-row X :outdegree
                                :reverse-outdegree)
((7 12) (11 13) (8 16) (15 17) (2 19)
 (18 20) (3 4) (9 10) (5 6))
```

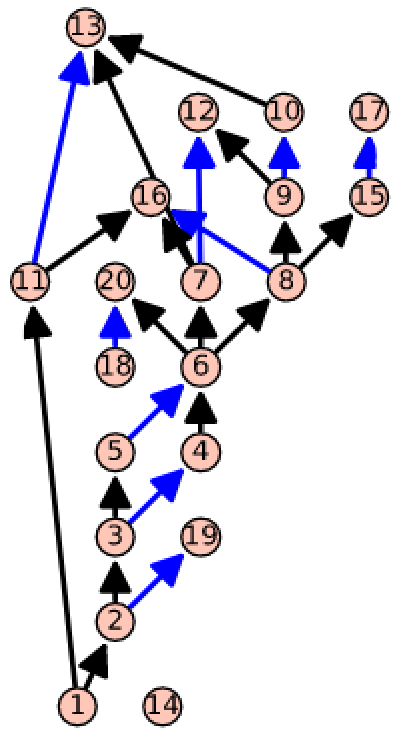


Figure 5.5: Discrete vector field on X (in blue) by using *:outdegree* for columns and *:reverse-outdegree* for rows.

Then, it is clear that the orderings chosen to move through the column and row indexes influence the computation of discrete vector fields on finite T_0-spaces. Also, it seems that the best strategies applied in a particular finite space might not be the best for other spaces. In the next section we will apply the strategies described above to compute discrete vector fields on random generated finite spaces, in order to evaluate and compare their performances.

5.2 Evaluating strategies on random finite spaces

The strategies considered above have been tested in random finite T_0-spaces. For a given dimension `n` and a given density `d`, we have generated random spaces and, on each one of them, we have computed 36 discrete vector fields in Kenzo by using the strategies *:standard*, *:random*, *:indegree*, *:reverse-indegree*, *:outdegree* and *:reverse-outdegree* (shorted as *s*, *r*, *i*, *ri*, *o* and *ro*, respectively) in order to sort the column and row indexes of the Stong matrices. For simplicity, we have used the symbol α - β to indicate that the strategy α was used to sort the columns and β was used to sort the rows of the Stong matrix of the corresponding finite T_0-space.

In Table 5.1, for each pair `n` (size) and `d` (density), 20 random finite T_0-spaces have been computed. In each cell, the five strategies that generated the longest discrete vector fields have been considered. Each parenthesis (m) in the cells of the table indicates that the corresponding strategy achieves the maximum cardinality, among the 36 computed discrete vector fields, in m of the 20 spaces. Observe that the strategy *o* - *o* predominates over the others when `n` increases and for low values of `d`. However, in most of the finite spaces we have considered, the different strategies have produced discrete vector fields with low difference between their lengths.

In Table 5.2, the greatest and smallest average length among the 36 discrete vector fields computed on 20 random finite T_0-spaces for each pair `n` (size) and `d` (density) are shown. In bold, the greatest average length corresponding to the strategies *o* - *o*, while the other results (the smallest) correspond to *ri* - *ro*, except the value marked with $(^*)$ which was obtained with *ro* - *ro*. Note that the percentage difference between the values in each cell of the table is less than 15%.

The data used for the application of the strategies are included at [CR20] in the folder /dvfields-strategies/results-length. In each of the folders found there, we have included a spreadsheet (.xlsx) where the lengths of the discrete vector fields obtained by using the 36 strategies are summarized.

It is important to remark that once we have modified the algorithm to compute discrete vector fields on finite T_0-spaces, any other sorting strategy can be defined and we can contrast with the above strategies in order to evaluate its performance, thanks to the function `dvfield-strategies-col-row`, which takes as parameters the sorting strategies on columns and rows.

Table 5.1: Comparison of the number of times that a strategy α - β achieves the maximum cardinality among the 36 strategies applied to random finite T_0-spaces.

n d	100	150	200	250	300	350
0.2	***o - o* (11)**	***o - o* (12)**	***o - o* (14)**	***o - o* (13)**	***o - o* (8)**	***o - o* (17)**
	o - i (10)	*o - i* (6)	*o - i* (7)	*o - i* (5)	*o - i* (7)	*o - i* (3)
	o - s (4)	*o - r* (4)	*o - r* (4)	*o - r* (4)	*o - s* (5)	*o - s* (2)
	o - r (4)	*o - s* (3)	*o - ri* (3)	*o - ro* (2)	*o - r* (3)	*o - ri* (2)
	o - ri (2)	*o - ro* (2)	*o - s* (1)	*o - s* (2)	*o - ri* (2)	*o - i* (1)
0.4	***o - o* (9)**	***o - i* (9)**	***o - o* (13)**	***o - o* (9)**	***o - o* (12)**	***o - o* (16)**
	o - i (8)	*o - o* (8)	*o - i* (10)	*o - i* (9)	*o - i* (9)	*o - i* (6)
	o - r (5)	*o - s* (3)	*o - s* (3)	*o - r* (3)	*o - r* (2)	*o - r* (1)
	o - s (3)	*o - r* (2)	*s - o* (3)	*s - o* (1)	*o - s* (2)	*s - o* (1)
	s - o (3)	*i - o* (2)	*o - r* (2)	*s - r* (1)	*o - ri* (1)	*i - ri* (1)
0.6	***o - r* (7)**	***o - o* (9)**	***o - i* (8)**	***o - o* (12)**	***o - o* (10)**	***o - o* (12)**
	o - o (6)	*o - i* (5)	*o - o* (6)	*o - i* (7)	*o - i* (7)	*o - i* (8)
	o - i (5)	*s - o* (5)	*s - o* (3)	*o - r* (3)	*s - o* (4)	*o - s* (3)
	o - s (5)	*o - s* (4)	*o - r* (3)	*o - s* (1)	*o - r* (3)	*o - r* (2)
	s - r (4)	*o - r* (2)	*r - o* (2)	*s - o* (1)	*o - ri* (1)	*s - o* (1)
0.8	***s - o* (9)**	***o - o* (10)**	***o - o* (8)**	***o - o* (13)**	***o - o* (8)**	***o - o* (9)**
	o - o (7)	*o - i* (6)	*s - o* (7)	*o - i* (8)	*o - i* (8)	*o - i* (8)
	o - s (4)	*s - o* (3)	*o - i* (7)	*s - o* (7)	*s - o* (7)	*s - o* (7)
	s - i (4)	*o - r* (3)	*r - o* (2)	*o - r* (4)	*o - r* (2)	*o - r* (2)
	s - s (4)	*i - o* (3)	*s - ri* (1)	*s - r* (2)	*o - s* (1)	*s - ri* (1)

Table 5.2: Comparison of the greatest and the smallest average length of the discrete vector fields computed on random finite T_0-spaces by using the 36 described strategies.

n d	100	150	200	250	300	350
0.2	**38,85**	**57,50**	**76,35**	**91,95**	**99,25**	**121,35**
	33,15	49,40	65,95	80,20	87,45	104,85
0.4	**40,85**	**64,10**	**83,30**	**106,55**	**123,50**	**147,90**
	35,85(*)	55,70	72,50	92,40	107,40	127,55
0.6	**44,50**	**67,30**	**88,70**	**110,95**	**124,85**	**156,50**
	38,85	58,20	77,50	96,40	110,00	136,30
0.8	**43,95**	**67,30**	**91,25**	**113,45**	**137,95**	**153,25**
	38,95	58,95	79,85	100,00	120,05	133,45

5.3 Machine learning strategies

In the previous two sections, some geometric and random strategies to compute discrete vector fields on finite topological spaces have been introduced. Moreover, the results that have been obtained by applying them on different random samples of finite topological spaces have been compared. As has been mentioned, it seems clear that, in most cases, using *out-degree* strategies helps obtaining discrete vector fields containing the greatest number of vectors. Nevertheless, it is also observed that it is not true when computations are carried out on spaces of small size and high density. It thus seems reasonable to use some other kind of methods and techniques which can be able to obtain vector fields as big as possible in all cases. In particular, in this section, *machine learning* methods are going to be applied. This kind of techniques has been successfully applied on a large variety of areas such as computer vision [KAH20], games [SHM16] or medical image treatment [MSLR19]. Some machine learning methods have also been considered for some problems in computer algebra [BLP20], [PSHL20], [Sil19].

In fact, machine learning covers a wide range of different alternatives for how the learning process can be carried out. In particular, due to the type of problem we are studying, we are going to focus on the so-called *reinforcement learning*. The key point of this type of learning is to establish a suitable reward system. That is, when the process has reached a concrete state and it has to select a possible action in order to continue, a reward is applied, which depends on the profit obtained after the corresponding change of state.

In a first attempt to introduce machine learning techniques to obtain a discrete vector field as big as possible, we use the *Q-learning* algorithm [Wat89], which is a particular case of reinforcement learning. This algorithm requires the definition of *states* (the possible situations which can be encountered), *actions* (transitions from one state to another state) and (positive or negative) *rewards* received after each transition.

More concretely, the Q-learning algorithm is based on the construction of a *Q-table*, which is a matrix where we have a row for every state and a column for every action. It is first initialized to 0, and then values are updated after training, taking into account the rewards obtained. The process of *training* is done by iterating a sequence of actions starting from the initial state, combining *exploration* (choosing a random action) and *exploitation* (choosing actions based on already learned Q-values). For each state (each row in the table), the best action is supposed to be the one with the highest value in the corresponding column of the Q-table. See [Wat89] for more details.

In our problem, we have chosen to use the Python library for reinforcement learning *Gym* [Ope16]. We construct a new class called `DVFMatrixEnv` which implements the interface `gym.Env`, which represents a learning `environment`. This interface requires the definition of a set of `states` and a set of `actions`, and the implementation of four methods: `init`, `reset`, `render` and `step`. The most important method is `step`, which, for an input pair of state and action, returns a new state, a reward and a boolean variable called `done` which determines if the problem ends or not after applying the action.

The input data to construct an environment of type `DVFMatrixEnv` is the Stong matrix of the finite T_0-space where we want to compute a discrete vector field. For instance, we consider

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

which is the Stong matrix of the finite T_0-space in Figure 5.6.

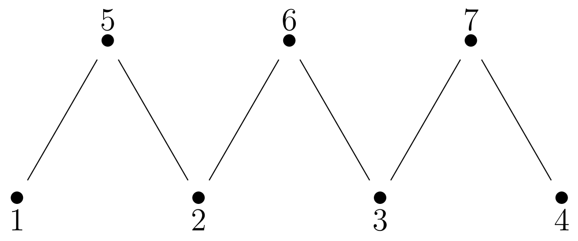


Figure 5.6: Finite T_0-space.

The elements of the set of actions are the possible vectors we can select in the matrix, which are coded as integers with the following formula:

$$\text{action} = \text{row} * \text{numberOfRows} + \text{column}$$

In our example there are 6 possible vectors:

$$[0*7+4, 1*7+4, 1*7+5, 2*7+5, 2*7+6, 3*7+6] = [4, 11, 12, 19, 20, 27]$$

Each state is given by a list of vectors, which is coded in binary form with length equal to the number of actions. For instance:

$$\begin{aligned} 0 &\leftrightarrow 000000 = \{\} \\ 8 &\leftrightarrow 000100 = \{19\} \\ 17 &\leftrightarrow 100010 = \{4, 20\} \end{aligned}$$

The number of possible states is $2^{\text{numberOfActions}}$. Then, a state represents a set of possible vectors (some of the sets are not valid, but this will be specified in the algorithm by means of a negative reward). The initial state is 0.

Given a state and an action, the method `step` proceeds as follows:

- The binary representation of the state (a list) is determined.

- The row and the column of the action are determined.
- For each vector in the state (that is, each element in the binary list which is equal to 1), the row and the column are computed.
- If any of the rows or columns which have been computed in the previous step are equal to those of the action, then the action is not valid and we take the reward equal to -1, the next state equal to the actual one and `done = True`.
- We check if the new vector can be added, so that we have a Morse matching (which means that there are no cycles in the modified Hasse diagram). If this condition is not fulfilled, we also consider that the action is not valid and we take again the reward equal to -1, the next state equal to the actual one and `done = True`.
- If the action is valid, then we compute the new state by adding the new vector (action):

  $$\text{newState} = \text{oldState} + 2^{\text{indexOfAction}}$$

  The reward in this case is equal to +1. We determine also if it is possible to add a new (valid) vector to the list; if it is not possible then we set `done=True`.

We render a state by its number and by a matrix where there are 1's in the positions of the selected vectors, and 0's elsewhere.

For instance, let us consider the environment associated to the matrix M. The number of actions (possible vectors) is 6 and the number of states is $2^6 = 64$.

```
env = DVFMatrixEnv(M)
print(env.num_actions)
6
print(env.num_states)
64
```

The initial state 0 is rendered as a 7×7 matrix of zero entries:

```
env.render()
0
[[0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0]]
```

Let us consider the action with index equals 2, that is, the third vector in the list of possible vectors which is the one with row equals 1 and column equals 5 (taking into account that indexes in lists and matrices in Python start in 0), that is, the vector $(2, 6)$ in Figure 5.6. We apply the action and render the new state, which is equal to 4.

```
action_index = 2
env.step(action_index)
env.render()
```

```
4
[[0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0]]
```

If we try to add a non-valid vector (for instance, $(3,5)$ in Figure 5.6, which has index 3 in the list of vectors), the state is not modified.

```
action_index = 3
env.step(action_index)
env.render()
4
[[0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0]]
```

Once we have defined our environment, we perform the training of the Q-learning algorithm with a given number of iterations. The result is obtained by means of the Q-table, which in this case is a 64×6 matrix. Let us observe the first 3 rows of this table:

```
q_table[:3,:]
[[ 1.75   1.75   1.75 1.75 1.75 1.75]
 [-0.25 -0.25   1.5  1.5  1.5  1.5]
 [-0.25 -0.25 -0.25 1.5    1    1.5]]
```

The first row corresponds to the initial state, and the columns are the aggregated rewards for each one of the possible actions. Let us observe that, in this row, all columns have the same value, which means that all the possibilities are *good*. The second row corresponds to the state 1, that is, having already selected the first vector $(1,5)$. In that situation, there are two negative values which correspond to vectors which are not valid (in this case, the same vector $(1,5)$ and the vector $(2,5)$), and the other 4 possibilities have the same positive value and therefore are *good*. The third row corresponds to the state 2, having selected the second vector $(2,5)$. In that case, there are three negative values which correspond to the three not valid vectors and then there are three positive values $1.5, 1$ and 1.5. This means that the fifth vector $(3,7)$ is worse than the fourth and sixth vectors $(3,6)$ and $(4,7)$. This is due to the fact that, if we select the vector $(3,7)$ together with the vector $(2,5)$, then all the other vectors are not valid, so that our discrete vector field would have only two vectors. However, if we select for instance the vector $(3,6)$ together with $(2,5)$, we can add also the vector $(4,7)$ obtaining in that way a discrete vector field with 3 vectors.

After a high number of iterations, the Q-table allows one to define a good process to obtain a discrete vector field as big as possible. We have implemented this process, which in our example produces the following discrete vector field with vectors $(1,5)$, $(2,6)$ and $(3,7)$, codified as 21. Let us observe in Figure 5.6 that its is (one of) the biggest possible vector fields.

```
computeBestDVF()
21
[[0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 1, 0],
 [0, 0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0]]
```

We have tested the method on some of the random examples studied in the previous section. In all the cases, the Q-learning technique has provided us satisfactory results. That is, for each space, the vector field we obtained by using this approach has the same number of vectors as the best one obtained by applying the strategies given in Section 5.2.

Nevertheless, the size of the Q-table results to be an evident problem to be treated. In fact, we only have tested satisfactorily the Q-learning technique in 13 spaces; for other spaces, the Q-table can not be built. It is a common drawback of this type of machine learning and there exist same well studied alternatives to avoid it. For instance, it is possible to use Monte-Carlo trees or neural networks instead of computing the whole Q-table.

Monte-Carlo tree search is a heuristic search algorithm for some kinds of decision processes, which has been employed in software that plays board games. The focus of Monte-Carlo tree search is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space. In our problem, each node of the tree is a state (sequence of vectors added to the discrete vector field). In a joint work with J. Divasón we have tested this method for some of the examples of the previous section and it works satisfactorily. Again, the vector field we obtained by using the Monte-Carlo tree search technique has the same number of vectors as the best one obtained by applying the strategies given in Section 5.2. Moreover, we can deal with bigger examples than the Q-learning method. Let us observe that in the Q-learning algorithm all the entries of the Q-table must be built, but in fact most of them are not been representing states (set of vectors) which can be reached. In contrast, by using Monte-Carlo tree search techniques, the corresponding method only generates random training chains of vectors which are valid. In this way, it is possible to deal with spaces having a bigger size than those that admit Q-learning techniques. The promising results obtained in our first attempts applying machine learning methods, open a new line of research where we plan to study the behavior of the algorithms over a bigger set of spaces and explore other techniques such as neural networks.

Conclusions and further work

In this work we have presented effective algorithms to compute invariants of finite topological spaces. These algorithms have been developed by combining combinatorial techniques on posets which have been stated in the foundational papers of the theory of finite spaces and recent results in this line of work, such us methods to maintain weak homotopy types or simple homotopy types and the application of Discrete Morse Theory.

Up to now, the known methods for computing invariants of finite topological spaces were applicable only for face posets of simplicial complexes or regular CW-complexes. In Chapter 2, some theoretical results on finite topological spaces (due to Stong, McCord, Barmak, Minian, Cianci and Ottina) have been made constructive, producing in particular new algorithms for computing in an explicit way some chain complexes associated with h-regular finite topological spaces (smaller than the chain complex of the order complex of the space) and their corresponding generators.

We have implemented the previous algorithms in the computer algebra system Kenzo. In this way, we can make use of Kenzo capabilities for dealing with matrices, chain complexes and homology. With our new algorithms, we endow Kenzo with new tools to deal with finite topological spaces. Up to our knowledge, our new program is the only software able to compute homology groups of finite topological spaces working directly on the posets without having to go, necessarily, to the simplicial world. Moreover, we improve our algorithms on h-regular spaces by using discrete vector fields. In our case, we produce a new algorithm constructing a discrete vector field defined directly on the poset that can be applied to general h-regular finite spaces; as before, up to our knowledge there does not exist any other software producing this kind of construction over general finite topological spaces.

The algorithms to compute homology above mentioned are applicable to h-regular finite spaces. In the literature there are few examples of h-regular finite spaces, different from face posets of simplicial complexes. The h-regularization process we have described in Chapter 3, produces a wide variety of h-regular finite spaces. Indeed, as we have shown, any finite T_0-space of height at most two can be h-regularized, allowing to consider new examples of this kind of spaces.

Some modifications to finite T_0-spaces had to be considered when we were searching for a correct implementation of h-regularization. In particular, when the tilded open minimal set of an element of height two is not connected, we *correct* this by introducing a beat point, which makes the subspace connected in the new space. The most challenging part of our method was Proposition 3.26, where we show an algorithmic way to *separate* the tilded open minimal set into 1-spheres, which was the key result to achieve the h-regularization of finite T_0-spaces that we had in mind. Some of the results we have presented can be applied to finite spaces with higher heights, then the search of effective methods to modify finite spaces of heights greater than two in order to satisfy the h-regular property has just started.

Moreover, the modifications we performed to h-regularize a finite T_0-space do not change the simple homotopy type and all of the spaces in the process are 3-deformations of the initial one. In the particular case when a homotopically trivial finite T_0-space of height two satisfies the Andrews-Curtis conjecture, all the spaces in the process of h-regularization satisfy it too, which could be used to attack the conjecture in a future work by using simple homotopy equivalent spaces to the given one as in [Fer17b], where some potential counterexamples to the conjecture have been discarded by means of techniques on finite topological spaces.

In Chapter 4, we have presented an interface between the computer algebra systems SageMath and Kenzo. Our work has made it possible to work with Kenzo in a friendlier way and to allow both systems to collaborate in some computations which can not be done independently in any of the programs. In addition, we have enhanced SageMath with new functionalities in algebraic topology, allowing the representation and computation of topological invariants of objects of infinite nature; this type of objects were not available in the system before. Since this work was carried out with open-source software, any user can explore and adapt the code, expecting to extend the use of Kenzo to a broader community.

Regarding to finite topological spaces, the module we have described in Section 4.3 uses all the functionalities described in this memoir by means of the interface between SageMath and Kenzo. It can be easily used by any user of SageMath, which makes different our work from the code found in [Fer17a] and [Ren19]. Of course, we intend to expand our module devoted to finite topological spaces with other algorithms developed by other authors and therefore complement the functionalities we have implemented. The tickets we have opened in the SageMath development process to include the functions described in Chapter 4 are already published, but they are subject to style changes and corrections proposed by the reviewers in order to be part of SageMath.

In Chapter 5, we have considered some strategies trying to study alternatives to compute longer discrete vector fields on finite spaces. Although there is not a unique strategy which performs well in all situations, we have shown an optimal strategy for most finite spaces with big size and small density. The machine learning techniques we have also explored have shown promising results, which allow us to continue investigating new methods and strategies that can be applied in the future, in order to compute other invariants and topological properties of finite spaces.

Appendix

In this appendix we have included the Lisp functions that have been used in the examples shown in our work, in order to make them reproducible. All the functions and methods described in this memoir are available at [CR20].

`binarymatrice-to-fbasis` *mtrx*
: If *mtrx* is the topogenous matrix of a finite T_0-space X, this function returns a vector whose n-th component is the list of the elements in the minimal open set of $n+1$ in the opposite space X^{op} i.e. F^X_{n+1} (remember that in Lisp the vectors are indexed beginning with the zero subscript).

`binarymatrice-to-ubasis` *mtrx*
: If *mtrx* is the topogenous matrix of a finite T_0-space X, this function returns a vector whose n-th component is the list of the elements in the minimal open set of $n+1$ i.e. U^X_{n+1}.

`cardinality` *finspace*
: It returns the number of elements in the space *finspace*.

`edges-to-stong` *dim edges*
: The parameter *edges* must be a list of pairs $(a\ b)$ representing the edges of the Hasse diagram of a finite T_0-space. The function returns the Stong matrix of dimension *dim* of the space associated to the poset given by *edges*.

`edges-to-stong-mtrx` *u-vector*
: The parameter *u-vector* is a vector whose n-th entry is the list of the elements in $\widehat{U}_{n+1}$. The function returns the Stong matrix representing the finite T_0-space induced by *u-vector*.

`h-regular-homology-sim` *finspace*
: It prints on the screen the homology groups of the h-regular space *finspace* from dimension 0 to the height of *finspace*.

`non-hausdorff-suspension` *finspace*
: It returns a `FINITE-SPACE` representing the non-Hausdorff suspension of the space *finspace*.

`show` *mtrx*

This function prints on the screen the matrix *mtrx* of type `matrice` or `array`.

`sorting-by-strategy` *finspace strategy*

It returns the list of elements of *finspace* sorted by using the ordering induced by *strategy*.

`stong-to-edges` *stong*

This is the inverse function of `edges-to-stong`: it takes a *stong* matrix and returns a list of pairs representing the edges of the Hasse diagram associated to *stong*.

Bibliography

[AC65] J. J. Andrews and M. L. Curtis, *Free groups and handlebodies*, Proceedings of the American Mathematical Society, American Mathematical Society **16** (1965), no. 2, 192–195.

[Ale37] P. Alexandroff, *Diskrete Räume*, Mat. Sb. (N.S.) **2** (1937), 501–518.

[Bar11] J. A. Barmak, *Algebraic topology of finite topological spaces and applications*, Lecture Notes in Mathematics, vol. 2032, Springer, 2011.

[BL17] B. Benedetti and F. H. Lutz, *Library of triangulations*, http://page.math.tu-berlin.de/~lutz/stellar/library_of_triangulations/, 2017.

[BLP20] Y. Bengio, A. Lodi, and A. Prouvost, *Machine learning for combinatorial optimization: a methodological tour d'horizon*, https://arxiv.org/pdf/1811.06128.pdf, 2020.

[CO17] N. Cianci and M. Ottina, *A new spectral sequence for homology of posets*, Topology and its Applications **217** (2017), 1–19.

[CO18] ______, *Poset splitting and minimality of finite models*, Journal of Combinatorial Theory, Series A **157** (2018), 120–161.

[CR16] J. Cuevas-Rozo, *Funciones submodulares y matrices en el estudio de los espacios topológicos finitos*, Tesis de maestría, Universidad Nacional de Colombia, 2016.

[CR18] ______, *Point reduction algorithms and discrete vector fields for finite topological spaces in the Kenzo system*, Proceedings of Fourth EACA International School on Computer Algebra and its Applications, 2018, https://www.usc.es/regaca/eacaschool18/files/Contributed_talks_EACA2018.pdf, pp. 3–4.

[CR19a] ______, *Cálculo de invariantes topológicos sobre espacios topológicos finitos*, XXII Congreso Colombiano de Matemáticas 2019, 2019, http://scm.org.co/wp-content/uploads/2019/06/Cronograma_XXII_CCM2019_V2.pdf.

[CR19b] ______, *h-regularización de espacios topológicos finitos*, VIII Encuentro de Jóvenes Topólogos, 2019, http://xtsunxet.usc.es/etop2019/EJTop2019-JCuevas.pdf.

[CR20] ______, *Finite topological spaces in Kenzo*, https://github.com/jcuevas-rozo/finite-topological-spaces, 2020.

[CRDMBR19] J. Cuevas-Rozo, J. Divasón, M. Marco-Buzunáriz, and A. Romero, *A Kenzo interface for algebraic topology computations in SageMath*, ACM Commun. Comput. Algebra **53** (2019), no. 2, 61–64.

[CRDMBR20] ______, *Integration of the Kenzo system within SageMath for new algebraic topology computations*, 2020, Preprint.

[CRLRS18] J. Cuevas-Rozo, L. Lambán, A. Romero, and H. Sarria, *New algorithms for computing homology of finite topological spaces*, Proceedings of 24th Conference on Applications of Computer Algebra - ACA 2018, 2018, pp. 108–112.

[CRLRS20a] ______, *Effective homological computations on finite topological spaces*, Applicable Algebra in Engineering, Communication and Computing (2020), In press.

[CRLRS20b] ______, *h-regularization of finite topological spaces*, 2020, Preprint.

[CRMBR19] J. Cuevas-Rozo, M. Marco-Buzunáriz, and A. Romero, *Computing with Kenzo from Sage*, Proceedings of the 2019 conference on Effective Methods in Algebraic Geometry, 2019, http://eventos.ucm.es/_files/_event/_12097/_editorFiles/file/Abstracts_MEGA-alphabetic.pdf, p. 23.

[Dev20] The SageMath Developers, *The Sage Mathematics Software System (Version 9.2)*, https://www.sagemath.org, 2020.

[DRSS99] X. Dousson, J. Rubio, F. Sergeraert, and Y. Siret, *The Kenzo program*, Institut Fourier, Grenoble, 1999, http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/.

[Fer17a] X. Fernández, *Finite-spaces*, https://github.com/ximenafernandez/Finite-Spaces, 2017.

[Fer17b] ______, *Métodos combinatorios y algoritmos en topología de dimensiones bajas y la conjetura de Andrews-Curtis*, Ph.D. thesis, Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales, 2017.

[Fer19] ———, *3-deformations of 2-complexes and Morse theory*, https://arxiv.org/pdf/1912.00115.pdf, 2019.

[For98] R. Forman, *Morse theory for cell complexes*, Advances in Mathematics **134** (1998), 90–145.

[Hat02] A. Hatcher, *Algebraic topology*, Cambridge University Press, 2002.

[Heb19] G. Heber, *A repackaged version of the Kenzo program by Francis Sergeraert and collaborators*, https://github.com/gheber/kenzo, 2019.

[Her11] J. Heras, *A Kenzo module computing homotopy groups*, 2011, https://esus.unirioja.es/psycotrip/archivos_documentos/homotopy.cl.

[JBADB17] M. W. Jahn, P. E. Bradley, M. Al-Doori, and M. Breunig, *Topologically consistent models for efficient big geo-spatio-temporal data distribution*, ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences **IV-4/W5** (2017), 65–72.

[KAH20] A. I. Khan and S. Al-Habsi, *Machine Learning in Computer Vision*, Procedia Computer Science **167** (2020), 1444–1451.

[Kov89] V. A. Kovalevsky, *Finite topology as applied to image analysis*, Computer Vision, Graphics and Image Processing **46** (1989), 141–161.

[Kri66] V. Krishnamurthy, *On the number of topologies on a finite set*, Amer. Math. Monthly **73** (1966), 154–157.

[LKE$^+$02] G. Liu, M. Kitazawa, M. Eguchi, Y. Fuwa, and Y. Nakamura, *Dilation and reduction processing in finite topological spaces and its application to inspection of printed boards*, Electronics and Communications in Japan (Part III: Fundamental Electronic Science) **85** (2002), no. 12, 89–100.

[May67] J. P. May, *Simplicial objects in Algebraic Topology*, Van Nostrand Mathematical Studies, University of Chicago Press, 1967.

[May20] ———, *Finite spaces and larger contexts (in progress)*, http://math.uchicago.edu/~may/REU2020/FINITEBOOK.pdf, 2020.

[MB15] M. Marco-Buzunáriz, *Kenzo*, https://github.com/miguelmarco/kenzo, 2015.

[McC66] M. C. McCord, *Singular homology groups and homotopy groups of finite topological spaces*, Duke Math. J. **33** (1966), no. 3, 465–474.

[Mil63] J. Milnor, *Morse theory. Based on lecture notes by M. Spivak and R. Wells topological spaces*, Annals of Mathematics Studies, Princeton University Press, Princeton **51** (1963), 153pp.

[Min12] G. Minian, *Some remarks on Morse theory for posets, homological morse theory and finite manifolds*, Topology and its Applications **159** (2012), no. 12, 2860–2869.

[MSLR19] A. Maier, C. Syben, T. Lasser, and C. Riess, *A gentle introduction to deep learning in medical image processing*, Zeitschrift für Medizinische Physik **29** (2019), no. 2, 86–101.

[Mun84] J. R. Munkres, *Elements of Algebraic Topology*, Addison Wesley Publishing Company, 1984.

[Ope16] OpenAI, *Gym*, https://gym.openai.com/, 2016.

[Pal20] J. Palmieri, *Examples of simplicial sets – Sage reference manual v9.2*, 2020, https://doc.sagemath.org/html/en/reference/homology/sage/homology/simplicial_set_examples.html#sage.homology.simplicial_set_examples.simplicial_data_from_kenzo_output.

[PSHL20] D. Peifer, M. Stillman, and D. Halpern-Leistner, *Learning selection strategies in buchberger's algorithm*, https://arxiv.org/pdf/2005.01917.pdf, 2020.

[Qui78] D. Quillen, *Homotopy properties of the poset of nontrivial p-subgroups of a group*, Adv. Math. **28** (1978), 101–128.

[Ren19] J. Renders, *Finite topological spaces in algebraic topology*, Project of Master of Science in Mathematics, Ghent University, 2019.

[Rot98] J. Rotman, *An introduction to algebraic topology*, Graduate Texts in Mathematics, vol. 119, Springer, 1998.

[RRS06] A. Romero, J. Rubio, and F. Sergeraert, *Computing spectral sequences*, Journal of Symbolic Computation **41** (2006), no. 10, 1059–1079.

[RS06] 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-MAP-2006-v3.pdf.

[RS10] A. Romero and F. Sergeraert, *Discrete Vector Fields and Fundamental Algebraic Topology*, https://arxiv.org/pdf/1005.5685.pdf, 2010.

[RSS97] J. Rubio, F. Sergeraert, and Y. Siret, *EAT: Symbolic Software for Effective Homology Computation*, Institut Fourier, Grenoble, 1997, https://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/EAT-program.zip.

[Ser94] F. Sergeraert, *The computability problem in Algebraic Topology*, Advances in Mathematics **104** (1994), no. 1, 1–29.

[Shi68] M. Shiraki, *On finite topological spaces*, Rep. Fac. Sci. Kagoshima Univ. **1** (1968), 1–8.

[SHM16] D. Silver, A. Huang, and C. Maddison, *Mastering the game of Go with deep neural networks and tree search*, Nature **529** (2016), 484—-489.

[Sil19] L. R. Silverstein, *Probability and Machine Learning in Combinatorial Commutative Algebra*, Ph.D. thesis, University of California Davis, 2019.

[Sto66] R. E. Stong, *Finite topological spaces*, Trans. Amer. Math. Soc. **123** (1966), no. 2, 325–340.

[Wat89] C. J. C. H. Watkins, *Learning from Delayed Rewards*, Ph.D. thesis, Cambridge University, 1989.

[Whi50] J. H. C. Whitehead, *Simple homotopy types*, Amer. J. Math. **72** (1950), 1–57.

[Whi52] G. Whitehead, *Fiber spaces and the Eilenberg homology groups*, Proceedings of the National Academy of Science of the United States of America **38** (1952), no. 5, 426–430.