# UNIVERSIDAD COMPLUTENSE DE MADRID
## FACULTAD DE CIENCIAS MATEMÁTICAS

**TESIS DOCTORAL**

# CW-Decompositions of Plane Algebraic Curves and Milnor Fibers of Non-Isolated Quasi-Ordinary Singularities

Descomposiciones en CW-Complejo de Curvas Algebraicas Planas y Fibras de Milnor de Singularidades Cuasiordinarias no Aisladas

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

**Pablo Simón Isaza Peñaloza**

DIRECTORES

**Enrique Artal Bartolo**
**Pedro Daniel González Pérez**
**Jorge Carmona Ruber**

Madrid

Universidad Complutense de Madrid
Facultad de Ciencias Matemáticas

Ph. D. Thesis

---

# CW-Decompositions of Plane Algebraic Curves and Milnor Fibers of Non-Isolated Quasi-Ordinary Singularities

---

# Descomposiciones en CW-Complejo de Curvas Algebraicas Planas y Fibras de Milnor de Singularidades Cuasiordinarias no Aisladas

---

*Author:*

Pablo Simón Isaza Peñaloza

*Supervisors:*

Enrique Artal Bartolo
*Universidad de Zaragoza*

Pedro Daniel González Pérez
*Universidad Complutense de Madrid*

Jorge Carmona Ruber
*Universidad Complutense de Madrid*

September 2019

UNIVERSIDAD
COMPLUTENSE
MADRID

**DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR**

D./Dña. Pablo Simón Isaza Peñaloza,
estudiante en el Programa de Doctorado Investigación Matemática,
de la Facultad de Ciencias Matemáticas de la Universidad Complutense de Madrid, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

CW-Decompositions of Plane Algebraic Curves and Milnor Fibers of Non-Isolated Quasi-Ordinary Singularities

y dirigida por: Enrique Artal Bartolo
Pedro Daniel González Pérez
Jorge Carmona Ruber

**DECLARO QUE:**

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita.

Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada de conformidad con el ordenamiento jurídico vigente.

En Madrid, a 18 de septiembre de 2019

Fdo.:

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

# Table of Contents

# Acknowledgements

En primer lugar quiero agradecer a mis directores, Enrique Artal Bartolo, Jorge Carmona Ruber y Pedro Daniel González Pérez por sus enseñanzas, paciencia y dedicación. También a los Profesores José María Montesinos Amilibia, Alejandro Melle Hernández, y José Ignacio Cogolludo Agustín. A todos ellos debo la oportunidad de haber podido estudiar en este bello país. Su guía y apoyo han sido invaluables para mí, tanto en un sentido matemático como personal.

También debo un agradecimiento especial a la Universidad Complutense de Madrid, a la Universidad de Zaragoza y al Programa de Formación de Personal Investigador, que han hecho posible la realización de estos estudios.

De otra parte, quiero agradecer a mis amigos, mis compañeros de doctorado, mi compañera de despacho Adela Latorre Larrodé, y a todos quienes de alguna manera me han acompañado durante esta etapa de mi formación.

Por último, quiero dar gracias a mi padre, cuyo apoyo ha constituido el sustento fundamental de este proyecto. Un apoyo que ha superado los ámbitos material, moral y matemático. Este doctorado ha sido posible gracias a él.

## Abstract

This is a brief abstract that outlines the topics and contents of this work. The reader interested in a more detailed overview can skip directly to the introduction.

The braid monodromy is an invariant of algebraic curves that encodes strong information about their topology. Let $C$ be an affine algebraic plane curve, defined by a polynomial function $f$, and having a generic projection on the $x$ axis of $\mathbb{C}^2$. The braid monodromy of $C$ can be presented as a homomorphism

$$\rho : \pi_1(\mathbb{C}\backslash\{x_1, \ldots, x_m\}) \longrightarrow \mathcal{B}_n,$$

where $x_1, \ldots, x_m$ are the values of $x$ on which $f(x, y)$ have multiple roots, and $\mathcal{B}_n$ denotes the braid group of $n$ strands. If we see the curve as the image of a multivalued function $g$, the image under $\rho$ of a given loop is determined by the paths in $\mathbb{C}^2$ that $(x, g(x))$ follows when $x$ runs along the loop.

The braid monodromy has a long story and its development and applications has passed through the works of Zariski ([44, 45]), van Kampen ([16]), Moishezon and Teicher ([26, 27, 28, 29, 30, 31]), and Carmona ([9]) among many others ([11, 10, 19, 37, 2, 18, 3]).

A result by Carmona ([9]) shows that the braid monodromy of a curve $C$ determines the topology of the pair $(\mathbb{P}^2, \bar{C})$. He also provided a program that calculates the braid monodromy of a curve from its equation. However, it remained an open problem to find what this topology actually is. This is, given the braid monodromy of $C$, to find a description for the topology of $(\mathbb{C}^2, C)$ or $(\mathbb{P}^2, \bar{C})$.

In this work we provide such a presentation for the affine case. It consists of a regular CW decomposition of the pair $(\mathcal{D}, C \cap \mathcal{D})$, where $\mathcal{D}$ is a large enough polydisc in $\mathbb{C}^2$. The construction uses the presentation of the braid monodromy in the form of local braids and conjugating braids. In this presentation the local braids must be given as an ordered set of independent sub-braids, associated with different preimages of a critical value of a generic projection. The main theorem concerning the algebraic curves states the good definition of this decomposition (Theorem 1.18).

We also provide a program that, from the braid monodromy, calculates this CW complex explicitly. Since Carmona has already given a program that calculates the braid monodromy of a curve from its equation, it is possible, by using both programs, to calculate the CW decomposition from an equation of the curve. A second program turns this CW complex into a simplicial complex thin enough to take a regular neighborhood of the curve. Both programs are included in the appendices. The projective case is also briefly discussed.

On the other hand, the topological study of the singular points of complex hypersurfaces has as its cornerstone the work of John Milnor, presented in [25]. In this book he introduces a fibration, now known as the Milnor fibration, which is an essential aspect of the topology around these points. Two important invariants immediately derive from it: the Milnor fiber and the monodromy of the fibration.

The Milnor fiber of isolated singularities has been intensively studied and is well understood. For the non-isolated case, however, much less is known.

A feature of the Milnor fiber of the non-isolated surface singularities that has been the subject of considerable research in recent times is its boundary (see [21, 22, 34, 41]). Another aspect of the Milnor fiber of the non-isolated singularities that has been studied is its homotopy type (see [38, 42, 40, 32, 33, 12]). Some results exist for the case of quasi-ordinary singularities as well ([7, 13]). All of these results cover topological aspects or properties of the Milnor fiber of non-isolated singularities without directly addressing its topological type.

In this work we provide a combinatorial model of the compact Milnor fiber of the quasi-ordinary surface singularities with a single Puiseux pair. This model is built in the following way. Through a series of steps, we construct a CW decomposition of the pair $(D, C \cap D)$, where $D$ is a large enough polydisc, and $C$ is the discriminant curve of the Milnor fiber. Then, by means of a branched covering, we lift up this decomposition into a CW decomposition of the compact fiber (Theorem 3.11). Another model for the same fiber, as a cyclic gluing of four-dimensional balls along certain solid tori, is also given (Theorem 3.13).

This construction allows us to see the compact Milnor fiber as the preimage of the four dimensional ball under a series of branched coverings. By studying the deck transformations of these coverings, we are able to calculate the geometrical monodromy of the Milnor fibration (Theorem 5.1), and the complex homology groups of the compact Milnor fiber (Theorem 4.1). We also calculate the fundamental group (Theorem 5.3) and the homology groups (Theorem 5.4) of the compact Milnor fiber.

## Resumen


Este es un breve resumen que describe los temas y contenidos principales de este trabajo. El lector interesado en una descripción más detallada puede saltar directamente a la intriducción.

La monodromía de trenzas es un invariante de las curvas algebraicas que codifica fuerte información acerca de su topología. Sea $C$ una curva algebraica afín plana, definida por una función polinómica $f$, y con una proyección genérica en el eje $x$ de $\mathbb{C}^2$. La monodromía de trenzas de $C$ puede ser presentada como un homomorfismo

$$\rho : \pi_1(\mathbb{C} \backslash \{x_1, \ldots, x_m\}) \longrightarrow \mathcal{B}_n,$$

donde $x_1, \ldots, x_m$ son los valores de $x$ sobre los cuales $f(x, y)$ tiene raíces múltiples, y $\mathcal{B}_n$ denota el grupo de trenzas de $n$ hebras. Si vemos a la curva como la imagen de una función multivaluada $g$, la imagen bajo $\rho$ de un lazo dado está determinada por los caminos en $\mathbb{C}^2$ que sigue $(x, g(x))$ cuando $x$ recorre el lazo.

La monodromía de trenzas tiene una larga historia y su desarrollo pasa por los trabajos de Zariski ([44, 45]), van Kampen ([16]), Moishezon y Teicher ([26, 27, 28, 29, 30, 31]) y Carmona ([9]), entre muchos otros ([11, 10, 19, 37, 2, 18, 3]).

Un resultado de Carmona ([9]) muestra que la monodromía de trenzas de una curva $C$ determina la topología del par $(\mathbb{P}^2, \bar{C})$. Carmona además proporcionó un programa que calcula la monodromía de trenzas de una curva a partir de su ecuación. Sin embargo, permaneció abierto el problema de determinar en efecto esta topología. Esto es, dada la monodromía de trenzas de $C$, encontrar una presentación para la topología de $(\mathbb{C}^2, C)$ o $(\mathbb{P}^2, \bar{C})$.

En este trabajo proporcionamos tal presentación para el caso de curvas afines. La misma consiste en una descomposición CW regular del par $(\mathcal{D}, C \cap \mathcal{D})$, donde $\mathcal{D}$ es un polidisco suficientemente grande en $\mathbb{C}^2$. La construcción de dicha descomposición utiliza la presentación de la monodromía de trenzas como trenzas locales y trenzas conjugadas. En esta presentación las trenzas locales deben estar dadas como un conjunto ordenado de sub-trenzas independientes, asociadas a las diferentes preimágenes de un valor crítico de una proyección genérica. El teorema principal sobre las curvas algebraicas afirma la buena definición de esta descomposición (Teorema 1.18).

También proporcionamos un programa que, a partir de una monodromía de trenzas, calcula este CW complejo explícitamente. Dado que Carmona ya había provisto un programa que calcula la monodromía de una curva a partir de su ecuación, es posible, utilizando ambos programas, calcular el CW complejo a partir de una ecuación de la curva.

Un segundo programa transforma este CW complejo en un complejo simplicial suficientemente fino como para tomar una vecindad regular de la curva. Ambos programas están incluidos en los apéndices. El caso proyectivo también es brevemente discutido.

De otra parte, el estudio topológico de los puntos singulares de hipersuperficies complejas tiene como piedra angular el trabajo de John Milnor, expuesto en [25]. En este libro es introducida una fibración, ahora conocida como la fibración de Milnor, que es un aspecto esencial de la topología alrededor de estos puntos. Dos invariantes importantes se derivan inmediatamente de ella: la fibra de Milnor y la monodromía de la fibración.

La fibra de Milnor de las singularidades aisladas ha sido intensamente estudiada y es bien entendida. En el caso no aislado, sin embargo, se sabe mucho menos.

Un rasgo de la fibra de Milnor de las singularidades de superficie no aisladas que ha sido objeto de considerable investigación en los últimos tiempos es su frontera (ver [21, 22, 34, 41]). Otro aspecto de la fibra de Milnor de la singularidades no aisladas que ha sido estudiado es su tipo de homotopía (ver [38, 42, 40, 32, 33, 12]). Algunos resultados existen también para el caso para el caso de las singularidades quasi-ordinarias ([7, 13]). Todos estos resultados abarcan aspectos o propiedades topológicas de la fibra de Milnor de las singularidades no aisladas sin abordar directamente su tipo topológico.

En este trabajo proporcionamos un modelo combinatorio para la fibra de Milnor de las singularidades cuasi-ordinarias de superficie con un par de Puiseux. Este modelo es construido de la siguiente forma. A través de una serie de pasos, construimos una descomposición CW del par $(D, C \cap D)$, donde $D$ es un polidisco suficientemente grande, y $C$ es la curva discriminante de la fibra de Milnor. Entonces, por medio de cubiertas ramificadas, levantamos esta descomposición en una descomposición CW de la fibra compacta (Teorema 3.11). También proveemos otro modelo para la misma fibra como un pegado cíclico de bolas de dimensión cuatro a lo largo de ciertos toros sólidos (Teorema 3.13).

Esta construcción nos permite ver a la fibra compacta de Milnor como la preimagen de una bola de dimensión cuatro bajo una serie de cubiertas ramificadas. Estudiando las transformaciones de cubierta de este recubrimiento calculamos la monodromía geométrica de la fibración de Milnor (Teorema 5.1), y los grupos de homología complejos de la fibra de Milnor compacta (Teorema 4.1). También calculamos el grupo fundamental (Teorema 5.3) y los grupos de homología (Teorema 5.4) de la fibra de Milnor compacta.

# Introduction

This thesis is devoted to the topological study of two objects coming from algebraic geometry. These are, on the one hand, the embedding of a plane algebraic curve within the affine or projective space, and, on the other hand, the Milnor fiber of a quasi-ordinary surface singularity with a single Puiseux pair, along with the related monodromy action.

Let us consider the plane algebraic curves in the first place. The fundamental tool we use in the study of these curves is the braid monodromy.

The braid monodromy is an invariant of algebraic curves that encodes strong information about their topology. It can be thought in the following way. Let $f : \mathbb{C}^2 \longrightarrow \mathbb{C}$ be a polynomial function of the form

$$f(x, y) = y^n + a_1(x)y^{n-1} + \cdots + a_{n-1}(x)y + a_n(x),$$

where each $a_i$ is a polynomial in $x$ with complex coefficients. Let $C$ be the algebraic curve defined by $f$.

For any given fixed value $c$, the intersection of the line $x = c$ with $C$ consists of the roots of $f(c, y)$. It is easily seen that only on a finite number of values $x_1, \ldots, x_m$ of $x$ does $f(x, y)$ have multiple roots. Therefore, $f$ defines a multivalued function $g : \mathbb{C}\backslash\{x_1, \ldots, x_m\} \longrightarrow \mathbb{C}$, where $g(c)$ consists of the $n$ points where the line $x = c$ cuts the curve $C$.

Let us consider a loop $\gamma : [0, 1] \longrightarrow \mathbb{C}\backslash\{x_1, \ldots, x_m\}$. Then, as $x$ travels along $\gamma$, its image $g(x)$ describes $n$ paths inside of $\mathbb{C}^2$, given by $(\gamma(t), g(\gamma(t)))$, producing a braid of $n$ strands.

It can also be seen that homotopic loops give rise to homotopic braids, which allows us to define a homomorphism

$$\rho : \pi_1(\mathbb{C}\backslash\{x_1, \ldots, x_m\}) \longrightarrow \mathcal{B}_n,$$

where $\mathcal{B}_n$ denotes the braid group of $n$ strands. This homomorphism is called a *braid monodromy* for $C$. And since all the monodromies of a given curve are related by a simple set of transformations (conjugation by any given braid and Hurwitz moves, see [9] and

[3] for a definition), yielding equivalence classes, it is possible to talk about the braid monodromy of a curve.

On the other hand, any braid running between points $\{a_1, \ldots, a_n\}$ and $\{b_1, \ldots, b_n\}$ defines a permutation of $n$ elements, where $i$ is sent to $j$ if there is a strand of the braid running from $a_i$ to $b_j$. This defines a homomorphism $\phi : \mathcal{B}_n \longrightarrow \Sigma_n$. In particular, the braid $\rho(\gamma)$ defines a permutation among the $n$ points of $g(x)$, given by $\phi(\rho(\gamma))$. It is worth noticing that this permutation is exactly the image of $\gamma$ under the covering monodromy $\mu : \pi_1(\mathbb{C}\backslash\{x_1, \ldots, x_m\}) \longrightarrow \Sigma_n$ that describes $C$ as a covering of $\mathbb{C}$ branched over $\{x_1, \ldots, x_m\}$. Therefore,

$$\phi \circ \rho = \mu.$$

This allows us to see that the braid monodromy includes all the information contained in the covering monodromy, but adds yet further information. Hence, the braid monodromy is a stronger invariant than the covering monodromy classically used to describe the Riemann surface associated with $g$.

The idea of the braid monodromy has its origin in the foundational article [44] by Oscar Zariski about the fundamental group of the complement of an algebraic curve. A well known theorem by Riemann ([44, p. 306]) states that given points $x_1, \ldots, x_m$ in $\mathbb{C}$, and permutations $\sigma_1, \ldots, \sigma_m$ assigned to $x_1, \ldots, x_m$ generating a transitive group, there exists an algebraic multivalued function $y(x)$, the branches of which are permuted according to $\sigma_i$ by surrounding the corresponding $x_i$ on a sufficiently small loop.

A more modern perspective allows us to observe that, since the fundamental group of $\mathbb{C}\backslash\{x_1, \ldots, x_m\}$ is free, and generated by loops around each of the points $x_1, \ldots, x_m$, then $\sigma_1, \ldots, \sigma_m$ define a covering monodromy $\mu : \pi_1(\mathbb{C}\backslash\{x_1, \ldots, x_m\}) \longrightarrow \Sigma_n$. The theorem states therefore that there exists an algebraic curve $C$ in $\mathbb{C}^2$, satisfying that the branched covering consisting of the projection of $C$ into the $x$ axis branches along $\{x_1, \ldots, x_m\}$, and that the monodromy of this covering is $\mu$.

In his article, Zariski addresses the generalization of this problem into two dimensions. In this case, we need to consider an algebraic plane curve $C$ and a permutation assigned to each generator of $\pi_1(\mathbb{C}^2\backslash C)$. Then we inquire about the existence of an algebraic function $z(x, y)$ branching along $C$, and such that its branches are permuted, by travelling along the generators of $\pi_1(\mathbb{C}^2\backslash C)$, according to the corresponding permutation.

A previous result by Enriques ([11]) implies that such a function exists, provided that all the relations among the generators of $\pi_1(\mathbb{C}^2\backslash C)$ are satisfied by its corresponding permutations. This is, provided that the assigned permutations define a covering monodromy $\mu : \pi_1(\mathbb{C}^2\backslash C) \longrightarrow \Sigma_n$.

This result was not a complete solution of the initial problem, though, because the referred relations were at that time unknown. Zariski's interest in [44] centers then on the problem of finding a method to calculate the fundamental group of the complement of an algebraic curve.

The Lefschetz Hyperplane Section Theorem, or Zariski-Lefschetz Theorem, which was already known, implied that given a curve $C$ and a generic vertical line $L$, the generators $g_1, \ldots, g_n$ of $\pi_1(L\backslash C)$ were generators of $\pi_1(\mathbb{C}^2\backslash C)$. Zariski's idea was to move these loops in $\mathbb{C}^2\backslash C$, along a continuous path of vertical lines, surrounding the singular points of $C$ and returning to $L$. By doing so, each loop $g_i$ was transformed into a loop $g_i'$ and, by reading $g_i'$ in terms of $g_1, \ldots, g_n$ in $L$, relations for $\pi_1(\mathbb{C}^2\backslash C)$ could be obtained. The braid monodromy was implicit here, because the transformation of $g_i$ into $g_i'$ is determined by the braid corresponding to the loop around the singular point into consideration.

In the same article, Zariski pointed out, though not completely proved, that the fundamental group of the complement of a sextic with six cusps over a conic is $\mathbb{Z}/2 * \mathbb{Z}/3$. He showed moreover that if the complement of a sextic with six cusps has this fundamental group, then the cusps must lie on a conic. Later investigations in [45] showed the likely existence of sextics with six cusps not lying on a conic, implying the possible existence of what it is now known as a Zariski pair, which can be thought as two curves that have the same topology but different embeddings in the projective space. Such findings greatly motivated the study of the complement spaces of algebraic curves. The first confirmed examples of Zariski pairs were found by Mutsuo Oka ([37]) and Enrique Artal ([2]).

Concerning the fundamental group of the complement of a curve, Zariski's arguments were rather informal, for which he asked Egbert van Kampen to give a rigorous topological proof of his results. It was he who in [16] described the method to find such a group, which is now known as the Zariski-van Kampen method, in its full generality. This method allowed to confirm the fundamental groups previously computed by Zariski.

As we have seen, the method relied on the process of moving a vertical line along a closed path, and thus sending the set of intersecting points of the line with the curve to itself. While Zariski and van Kampen spoke only about permutations in this regard, this process actually yields a richer object, a braid, which carries not only all the information of the permutation, but also the information about how the points are permuted by travelling through the space. Oscar Chisini seems to have been the first one to realize about the importance of this fact, which led him to define the braid monodromy in [10].

The braid groups, although already implicit in previous works, were explicitly introduced and studied by Emil Artin in [4, 5, 6]. The theory developed in these articles provided an adequate setting for the braid monodromy's definition and technique, being of particular importance the introduction of a convenient algebraical presentation. It is easy to see that the braid group of $n$ strands defines an action on the fundamental group of a complex line punctured at $n$ points. The perspective opened by Chisini's approach allows us to see that the Zariski-van Kampen relations are given by the action of $b$ on the generators $g_1, \ldots, g_n$, where $b$ is the image by the braid monodromy of the loop surrounding the projection of the singular point into consideration.

Some decades later the braid monodromy was used by Boris Moishezon on the study of projective surfaces. In [26], he considers a projective surface as a covering of the projective plane branched along a discriminant curve, and uses the braid monodromy of the curve to obtain results about the surface. A systematic study of the braid monodromy was continued by him and Mina Teicher in [27, 28, 29, 30, 31], where they applied it to diverse problems.

Later on, Anatoly Libgober showed in [19] that the braid monodromy of an affine plane curve determines not only the fundamental group, but furthermore the homotopy type of its complement.

It was shown later the even stronger fact that the braid monodromy of a curve $C$ determines the topology of the pair $(\mathbb{P}^2, C)$. This was first proved by Kulikov and Teicher in [18] for the particular case of curves having only nodes and cusps. The full general case was proved by Jorge Carmona in [9], by using arguments that rely heavily on the graph manifold structure of certain neighborhoods (see [43, 35]). There, he also provided a program that calculates the braid monodromy of a curve from its equation.

Although by these results it became known that the braid monodromy determines the topology of $(\mathbb{P}^2, C)$, it remained an open problem to find what this topology actually is. This is, given the braid monodromy of an affine or projective curve $C$, to find a presentation for the topology of $(\mathbb{C}^2, C)$ or $(\mathbb{P}^2, \bar{C})$.

In this work we provide such a presentation for the affine case. The presentation consists of a regular CW decomposition of the pair $(\mathcal{D}, C \cap \mathcal{D})$, where $\mathcal{D}$ is a large enough polydisc in $\mathbb{C}^2$. This is, a regular CW decomposition of $\mathcal{D}$ having $C \cap \mathcal{D}$ as a subcomplex. The construction uses the presentation of the braid monodromy in the form of local braids and conjugating braids. This presentation must satisfy that the local braids can be expressed as an ordered set of independent sub-braids, associated with different preimages of a critical value of a generic projection. From here, we construct certain balls $T_0, \ldots, T_m$ and $B_1, \ldots, B_m$ associated with the local braids and the conjugating braids respectively. Each of this balls is embedded in $\mathbb{C}^2$, and has a CW decomposition such that the intersection of the ball with $C$ is a subcomplex. By joining all of these balls, we obtain a CW complex that decomposes $(\mathcal{D}, C \cap \mathcal{D})$. Theorem 1.18 states the good definition of this decomposition.

We also provide a program that, from the braid monodromy, calculates this CW complex explicitly. Since Carmona has already given a program that calculates the braid monodromy of a curve from its equation, it is possible, by using the two programs, to calculate the CW decomposition of $(\mathcal{D}, C \cap \mathcal{D})$ from an equation of the curve. A second program turns this CW complex into a simplicial complex thin enough to take a regular neighborhood of the curve. The projective case is also briefly discussed.

It is yet unknown if two topologically equivalent curves have the same braid monodromy, though a partial converse was proved by Artal, Carmona and Cogolludo in [3].

Let us consider now the Milnor fiber of a quasi-ordinary surface singularity with a single Puiseux pair. The topological study of the singular points of complex hypersurfaces has as its cornerstone the work of John Milnor, presented in his book [25]. In this book he introduces a fribration, now known as the Milnor fibration, which is an essential aspect of the topology around these points.

Let $f : (\mathbb{C}^{n+1}, 0) \longrightarrow (\mathbb{C}, 0)$ be a hypersurface singularity germ. If this singularity is isolated, then there exists $\varepsilon > 0$ such that, for every $\varepsilon'$ with $0 < \varepsilon' \leq \varepsilon$,

$$f^{-1}(0) \pitchfork S_{\varepsilon'},$$

where $S_{\varepsilon'}$ denotes the sphere centered at the origin with radius $\varepsilon'$. If the singularity is not isolated we have a similar property. In this case there exists a stratification of $f^{-1}(0)$ such that each stratum is transversal to $S_{\varepsilon'}$.

For this $\varepsilon$, there exists $\eta \ll \varepsilon$ such that

$$f^{-1}(z) \pitchfork S_\varepsilon$$

for every $z$ with $0 < |z| \leq \eta$. Let $D^*_\eta$ be the complex disk of radius $\eta$ punctured at the origin, and $B_\varepsilon$ the closed ball of radius $\varepsilon$ in $\mathbb{C}^{n+1}$. Let $X_{\varepsilon,\eta}$ be defined by $X_{\varepsilon,\eta} = B_\varepsilon \cap f^{-1}(D^*_\eta)$. Then,

$$f\,|_{X_{\varepsilon,\eta}} \colon X_{\varepsilon,\eta} \longrightarrow D^*_\eta$$

is a locally trivial fiber bundle, which is independent of $\varepsilon$ and $\eta$. This fiber bundle is called the *Milnor fibration* of the singularity.

Two important invariants immediately derive from here. On the first hand, the fiber $F$ of the fibration, which is given by the preimage of any point in $D^*_\eta$. This fiber is an analytic manifold with boundary called the (compact) *Milnor fiber*.

On the other hand we have a monodromy. A trivialization of the Milnor fibration yields a diffeomorphism $\rho : F \longrightarrow F$ defined up to isotopy. This map expresses how the fiber is taken into itself by travelling along $\partial D^*_\eta$ and completing a loop. This diffeomorphism is called the *geometric monodromy* of the fibration, and completely determines it. The homomorphisms $\rho_* : H_*(F;\mathbb{Z}) \longrightarrow H_*(F;\mathbb{Z})$ that it induces are also important invariants called *algebraic monodromies*.

For the case of isolated singularities, Milnor proved that the Milnor fiber has the homotopy type of a wedge sum or bouquet of $n$-dimensional spheres. The number $\mu$ of these spheres is called the Milnor number of the singularity, and is computable from the expression of $f$. Following these results, the Milnor fiber of this kind of singularities has been intensively studied and is, by now, well understood. For the non-isolated case, however, much less is known.

A feature of the Milnor fiber of the non-isolated surface singularities that has been the subject of considerable research in recent times is its boundary. In the study of the isolated singularities, the link of the singularity, which is also the boundary of the Milnor fiber, plays a central role. In the case of non-isolated singularities, however, this link is not smooth and the study of the boundary of the Milnor fiber (which is smooth) proves to be a natural path to follow. In [21, 22] Françoise Michel and Anne Pichon showed that the boundary of the Milnor fiber of a surface singularity with one-dimensional critical locus is a graph manifold ([43, 35]). In [34] András Némethi and Ágnes Szilárd obtained the same result by different methods, and developed an extensive study of the boundary of the Milnor fiber of this kind of singularities. Homological results were found by Dirk Siersma in [41].

Another aspect of the Milnor fiber of the non-isolated singularities that has been studied is its homotopy type. On this matter several Milnor style bouquet theorems have been given for certain families of singularities by Siersma and Némethi ([38, 42, 40, 32, 33]), and more recently by J. Fernández de Bobadilla and Miguel Marco ([12]).

Some results exist for the case of quasi-ordinary singularities as well. In [7], Chunsheng Ban, Lee McEwan and Némethi showed that the Euler characteristic of the Milnor fiber, for an irreducible quasi-ordinary surface singularity $f$, is the Euler characteristic of the Milnor fiber of the plane curve singularity defined by $g(x, z) = f(x, 0, z)$ (provided an adequate coordinate system). A similar result about the zeta-function of a hypersurface quasi-ordinary singularity, and thus concerning the algebraical monodromies of its Milnor fiber, was proved in [13] by Pedro D. González, McEwan and Némethi.

All of these results cover topological aspects or properties of the Milnor fiber of non-isolated singularities without directly addressing its topological type. This is because of the great diversity and complexity of these spaces. The same reason often obstructs the extraction of general results, bounding the studies to be restricted to particular families of functions.

In this work we provide a topological model of the compact Milnor fiber of the singularities of type $z^n - x^a y^b$, i.e. quasi-ordinary surface singularities with a single Puiseux pair. This model is built in the following way. First, in the spirit of the first part, we construct a CW decomposition of the pair $(\mathcal{D}, C \cap \mathcal{D})$, where $\mathcal{D}$ is a large enough polydisc in $\mathbb{C}^2$, and $C$ is the discriminant curve of the Milnor fiber. Then, by means of a branched covering, we lift up this decomposition into a CW decomposition of the compact fiber. The good definition of this model is shown in Theorem 3.11. Another model for the same fiber, as a cyclical gluing of four-dimensional balls, along certain solid tori, is also given in Theorem 3.13.

This construction allows us to see the compact Milnor fiber as the preimage of the four dimensional ball under a series of branched coverings. By studying the deck transformations of these coverings, we are able to calculate the geometrical monodromy of the Milnor fibration. This result is stated in Theorem 5.1.

The action of these deck transformations on the Milnor fiber, seen as a CW complex, also induces transformations on the complex chain spaces of the fiber. In fact, it induces a module structure. This allows us to decompose the complex chain spaces as a direct sum of the eigenspaces of the operators induced by the deck transformations. By studying the behaviour of the boundary operators within these eigenspaces we are able to calculate the complex homology groups of the compact Milnor fiber, which are provided in Theorem 4.1.

Also, by using the classical Zariski-van Kampen method we calculate the fundamental group of the complement of the curve $xy(xy - 1) = 0$. From here, by using covering theory, we are able to calculate the fundamental group of the compact Milnor fiber, given in Theorem 5.3. Finally, by using the previous results and the Universal Coefficient Theorem for Homology, we calculate the homology groups of the compact Milnor fiber, which are provided in Theorem 5.4.

We finish this introduction by describing the structure of the work, which is as follows.

In Chapter 1 we provide a regular CW decomposition of the pair $(\mathcal{D}, C \cap \mathcal{D})$, where $C$ is an affine plane curve and $\mathcal{D}$ is a large enough polydisc. This decomposition is obtained by successively building decompositions of pairs of spaces of increasing complexity. A section is dedicated to each of these pairs. In Section 1.1 we give preliminary definitions.

In Section 1.2 we construct a decomposition of a cylinder containing a braid. In Section 1.3 we do the same for the cone of a three-dimensional sphere that contains a closed braid. In Section 1.4 we give a decomposition for a small ball around a point of a curve. In Section 1.5 we decompose certain sets associated with the local braids. In Section 1.6 we define a certain complex that we use later to join the different complexes we obtain. In Section 1.7 we decompose sets associated with the conjugating braids. Finally, in Section 1.8, we glue all of these complexes to obtain a decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$.

In Chapter 2 we explain the programs for this decomposition and address the projective case. In Section 2.1 we explain the program that calculates the CW decomposition of $(D, \Omega \cap D)$, with the code included in appendix A. In Section 2.2 we explain the program that calculates the simplicial decomposition, with the code included in appendix B. The projective case is examined in Section 2.3.

In Chapter 3 we build the models for the compact Milnor fiber $\mathcal{CF}$ of the singularities of type $z^n - x^a y^b$. In Section 3.1 we construct a CW decomposition for the pair $(D, C \cap D)$, where $C$ is the curve with equation $xy(xy-1) = 0$ and $D$ a large enough polydisc. By lifting this decomposition through a branched covering, in Section 3.2 we obtain a similar decomposition for $(\mathcal{D}, C' \cap \mathcal{D})$, where $C'$ is the curve with equation $x^a y^b - 1 = 0$. In Sections 3.3 and 3.4 we explain respectively the topology and combinatorics of this pair. In Section 3.5, by lifting once again in a similar fashion, we obtain a decomposition for the compact Milnor fiber. In Section 3.6 we explain the topology of the fiber, providing another topological model. In Section 3.7, we explain the combinatorics of the fiber seen as a CW complex.

In Chapter 4 we calculate the complex homology of the Milnor fiber $\mathcal{CF}$. In Section 4.1 we provide some preliminary definitions and lemmas. In Section 4.2 we show that the complex chain spaces can be decomposed in certain ways, and find convenient bases for them. In Section 4.3 we examine the behaviour of the boundary operators. Finally, in Section 4.4, we calculate the complex homology groups with the aid of a program contained in appendix C.

In Chapter 5 we calculate several invariants of the Milnor fiber and fibration of the singularities of type $z^n - x^a y^b$. In Section 5.1 we calculate the geometrical monodromy of the Milnor fibration. In Section 5.2 we calculate the fundamental group of the complement of the curve $xy(xy-1) = 0$. By using this group and covering techniques, in Section 5.3 we calculate the fundamental group of the Milnor fiber. Finally, in Section 5.4, we use the previous results to calculate the homology groups of the fiber.

# Chapter 1

# A CW Decomposition for an Affine Algebraic Plane Curve

In this chapter we will make extensive use of the braid monodromy and related concepts. For definitions and a detailed treatment of these topics we recommend the references [3] and [9].

As we have already stated in the introduction, Carmona showed that the braid monodromy of an algebraic curve $C$ determines the topology of the pair $(\mathbb{P}^2, C)$. This result uses the concept of equivalent monodromies, which means monodromies that can be obtained from one another by conjugation by any given braids and by Hurwitz moves. His theorem states the following.

**Theorem by Carmona** *Let $C_1$ and $C_2$ be two projective plane curves with equivalent braid monodromies. Let us suppose that the line at infinity is not tangent to either $C_1$ or $C_2$. Then $C_1$ and $C_2$ are ambient isotopic (*[9, Theorem 4.2.1]*).*

In this chapter we use the braid monodromy of a plane affine curve $C$ to provide a topological model of the pair $(\mathcal{D}, C \cap \mathcal{D})$, where $\mathcal{D}$ is a large enough polydisc. Aside from the boundary of $\partial\mathcal{D}$, this is the same as providing a model of $(\mathbb{C}^2, C)$. Therefore, what we give is a complete topological description of the embedding of $C$ into $\mathbb{C}^2$. The model we give consists of a CW decomposition of the pair $(\mathcal{D}, C \cap \mathcal{D})$, i.e., a CW decomposition of $\mathcal{D}$ having $C \cap \mathcal{D}$ as a subcomplex. Besides, this decomposition is regular.

## 1.1 Preliminaries

Let $\Omega$ be an algebraic curve defined by a polynomial function $f : \mathbb{C}^2 \to \mathbb{C}$. Then, we can assume that $f$ is of the form

$$f(x, y) = y^n + a_1(x)y^{n-1} + \cdots + a_{n-1}(x)y + a_n(x),$$

where, for every $i$, $a_i(x) \in \mathbb{C}[x]$ with $\deg(a_i(x)) \leq i$ or $a_i(x) \equiv 0$. This is so because, if $f$ were otherwise, we could make a change of variable $x \longmapsto x - cy$, where $c$ is a complex number such that $x - cy$ does not divide the homogeneous part of higher degree of $f$ (see [14, Lemma 2.7]). Since $x \longmapsto x - cy$ is a linear isomorphism, this operation does not alter the topology of $\Omega$. Also, we may assume that $f$ does not have multiple factors since, if it had them, they could be removed without altering its set of zeroes.

Let $\Delta$ be defined by

$$\Delta = \{x \in \mathbb{C} \mid f(x, y) \text{ has multiple roots}\}.$$

**Claim 1.1.** *The set $\Delta$ is finite.*

*Proof.* We know the general fact that if $A$ is a U.F.D., then $q \in A[t]$ has multiple irreducible factors if and only if the discriminant of $q$, $disc_t(q)$, is equal to zero. By taking $A = \mathbb{C}[x]$ we have that $f \in \mathbb{C}[x][y]$, as element of $A[y]$, has multiple factors if and only if $disc_y(f) = 0$. Let us notice that $disc_y(W)$ is a polynomial belonging to $\mathbb{C}[x]$, that we will call $D(x)$ and, since $f$ has not multiple factors, $D(x)$ is not identically zero.

Now, for any fixed $a$, $f(a, y)$ has multiple roots if and only if $D(a) = 0$. Hence, $\Delta = \{a \in \mathbb{C} \mid D(a) = 0\}$, which is a finite set of points. $\square$

Let $x_1, \ldots, x_m$ be the points of $\Delta$, and let $x_0$ be a point of $\mathbb{C} \setminus \Delta$. Also, let $\eta_1, \ldots, \eta_m$ be a geometric set of generators of $\pi_1(\mathbb{C} \setminus \Delta, x_0)$, this is, a set of generators satisfying that $\eta_1 \cdot \ldots \cdot \eta_m$ is homotopic to the boundary of a disk centered at infinity.

Each of these generators can be chosen to be of the form $\eta_i = \lambda_i \gamma_i \lambda_i^{-1}$, where $\gamma_i$ is a small loop around $x_i$, and $\lambda_i$ is a path going from $x_0$ to the initial point of $\gamma_i$, as shown in the following figure.

Let

$$\rho : \pi_1(\mathbb{C} \backslash \{x_1, \ldots, x_m\}) \longrightarrow \mathcal{B}_n$$

be the braid monodromy of $f$, presented as a homomorphism. Then, the image of each $\eta_i$ under this monodromy can be obtained as a conjugated braid of the form $\rho(\eta_i) = \alpha_i \beta_i \alpha_i^{-1}$, where $\beta_i$ and $\alpha_i$ are as follows.

The braid $\beta_i$ is given by the monodromy of $f$ around $\gamma_i$, taking the initial point of $\gamma_i$ as a base point. On the other hand, the braid $\alpha_i$ is a certain braid associated with the open path $\lambda_i$. We will see that, although the braid monodromy is not defined for open paths, there is in fact a way to associate a braid $\alpha_i$ to the open path $\lambda_i$. The definition of

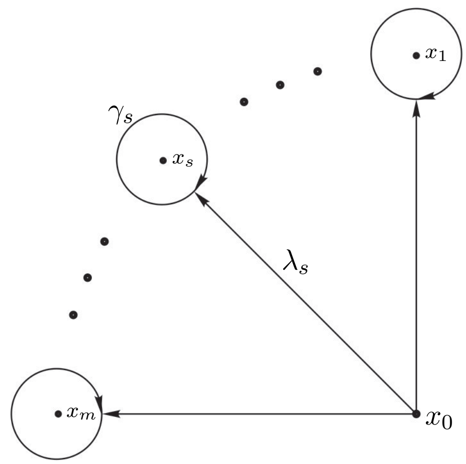


**Figure 1.1**

this braid is fairly similar to the one made for braids over closed paths, and is such that the decomposition $\rho(\eta_i) = \alpha_i \beta_i \alpha_i^{-1}$ holds.

We call $\beta_i$ and $\alpha_i$ in this context a *local braid* and a *conjugating braid* respectively. We will now take a closer look to these braids and provide precise definitions for them.

For any given point $z \in \mathbb{C}$ let us denote the complex line $x = z$ by $L_{x=z}$. We will use a similar notation for all the lines in $\mathbb{C}^2$, writing $L$ and the equation of the line as a subindex. Let us define

$$V_{x=z} := L_{x=z} \cap V(f).$$

Then, $V_{x=z}$ is a finite set with at most $n$ points, and exactly $n$ if $z \notin \Delta$. Let $a \in \mathbb{C} \backslash \Delta$ and let us choose a collection $\{\rho_i\}_{i=1}^{n-1}$ of simple curves sequentially connecting in $L_{x=a}$ the $n$ points of $V_{x=a}$, such that $\rho_i$ and $\rho_{i'}$ are always disjoint, except perhaps for their ends.

**Definition 1.1.** We call $\{\rho_i\}_{i=1}^{n-1}$ a *system of sequentially connecting paths for $f$ at $a$*, or an SCP for short.

This is what Moishezon called a skeleton in a slightly different context ([28]). Such a system always defines an isomorphism between the braid group $\mathcal{B}_n$ and the group of isotopy classes of homeomorphisms from the pair $(L_{x=a}, V_{x=a})$ into itself that fix a disk centered at infinity. We refer to this group as the mapping class group of $(L_{x=a}, V_{x=a})$ relative to the disk centered at infinity. The correspondence is defined in the following way.

For $1 \leq i \leq n-1$, let $\psi_i : L_{x=a} \longrightarrow L_{x=a}$ be a homeomorphism satisfying that $\psi_i(V_{x=a}) = V_{x=a}$ and consisting of a rotation by $180°$ of the disk that is a regular neighborhood of the curve $\rho_i$. Let us observe that this homeomorphism transposes the points of $V_{x=a}$ at both ends of $\rho_i$. The isotopy classes of the $\psi_i$ form a generating set for the mapping class group of $(L_{x=a}, V_{x=a})$ relative to the disk centered at infinity. Then, the

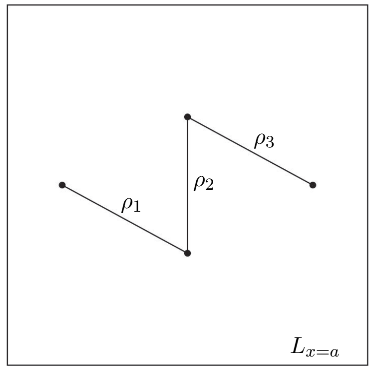


**Figure 1.2:** Example of an SCP for a set $V_{x=a}$ consisting of four points.

isomorphism is given by the correspondence of the class of each $\psi_i$ with the Artin generator $\sigma_i$ of $\mathcal{B}_n$, which is a half-twist between the $i$-th and $(i+1)$-th strands.

Let $\hat{x}$ be a point in $\mathbb{C}$, and let us consider a loop $\gamma$ defined by the parametrization $\gamma(t) = \hat{x} + \varepsilon e^{2i\pi t}$ $(t \in I)$, for some $\varepsilon > 0$. We choose $\varepsilon$ small enough so that no point of $\Delta$ is contained in the disk bounded by $\gamma$, with the possible exception of $\hat{x}$. For simplicity, we denote $\gamma(I)$ also by $\gamma$.

Let $V_\gamma$ denote the points of $V(f)$ with first coordinate in $\gamma$. Then, the pair

$$((\gamma \times \mathbb{C}), V_\gamma) = \left(\bigcup_{t \in I} L_{x=\gamma(t)}, \bigcup_{t \in I} V_{x=\gamma(t)}\right)$$

is naturally a fiber bundle pair over the base space $\gamma$. A trivialization of this bundle yields a homeomorphism $\varphi : L_{x=a} \longrightarrow L_{x=a}$ such that if we cut $((\gamma \times \mathbb{C}), V_\gamma)$ along $L_{x=a}$, and re-glue according to $\varphi$, the obtained space is the product of $(L_{x=a}, V_{x=a})$ by $S^1$.

Let $\beta$ be the braid corresponding to the isotopy class of $\varphi$. This braid is called the local braid of $f$ around $\hat{x}$, and can be thought as the braid formed by the $n$ points of $V_{x=\gamma(t)}$ as $t$ goes from 0 to 1. Thus, the link $\bar{\beta}$ obtained by closing $\beta$ is equal to $V_\gamma$, which is embedded in $(\gamma \times \mathbb{C}) \subset D^\delta_{x_0} \times \mathbb{C}$. We often think about $\beta$ realized in this way and not as an abstract braid.

Let us notice that, by taking $\varepsilon$ small enough, this local braid can be defined in the same way for $f \in \mathbb{C}\{x - \hat{x}\}[y]$ and for $f \in \mathbb{C}\{x - \hat{x}, y\}$, so we can speak about local braids in these contexts too.

Let us define now a braid associated to an open path. Let $\lambda$ be a simple curve in $\mathbb{C}\backslash\Delta$ with initial point $b$ and final point $a$. Let $\omega = \{\omega_i\}_{i=1}^{n-1}$ be an SCP at $b$ and $\varrho = \{\rho_i\}_{i=1}^{n-1}$ an SCP at $a$. Let $V_\lambda$ denote the points of $V(f)$ with first coordinate in $\lambda$. Then, by identifying $\omega$ and $\varrho$ with a straight line in the real part of $\mathbb{C}$, and the points of $V_{x=b}$ and $V_{x=a}$ with $1, \ldots, n$, we obtain $V_\lambda$ as a classically defined braid inside of $\lambda \times \mathbb{C}$, that we call $\alpha$.

Is worth noticing also that, by identifying $L_{x=b}$ and $L_{x=a}$ by a homeomorphism that sends each $\omega_i$ into $\rho_i$, it is also possible to obtain $\alpha$ as the braid corresponding to the isotopy class of a homeomorphism in the same way we did for $\beta$.

If $\lambda$ is one of the paths $\lambda_i$ defined before, we call $\alpha$ a conjugating braid for the corresponding $x_i$. Then, by choosing an SCP on $x_0$ and on the initial point of each $\gamma_i$ we obtain, for each $\gamma_i$ a local braid $\beta_i$, and for each $\lambda_i$ a conjugating braid $\alpha_i$. By assigning to each $\eta_i$ the conjugated element $\alpha_i\beta_i\alpha_i^{-1}$ we obtain a presentation for the braid monodromy of $f$.

This is the setting we use for constructing the CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$. As already explained in the introduction, we develop this construction by successively building CW decompositions of pairs of spaces of increasing complexity. We thus start by constructing a CW decomposition of a cylinder containing a braid (Section 1.2), followed by a decomposition of the cone of a three-dimensional sphere that contains a closed braid (Section 1.3), and then by decompositions associated with the local braids (Section 1.5), conjugating braids (Section 1.7), and finally the decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ (Section 1.8). The final decomposition is best understood by going from a local to a global perspective in this way, while thinking about it from the global to the local is only desirable retrospectively.

## 1.2 Decomposition of a Cylinder Containing a Braid

We begin by describing a CW decomposition of a torus with a closed braid embedded inside. Let us consider the points $P_j := (j, 0, 0)$ and $Q_j := (j, 0, c)$ of $\mathbb{R}^3$, where $j \in \{1, \dots, n\}$ and $c$ is a natural number to be defined. For each $j$, let $h'_j$ be a polygonal or smooth path, with strictly monotonous third coordinate, joining $P_j$ with $Q_j$. Let us suppose that the paths $\{h'_j\}$ are disjoint. Then $h'_1, \dots, h'_n$ constitute a braid $b$ of $n$ strands.

Let $\mathcal{B}_n$ be the braid group for $n$ strands. Let $e$ be the identity of this group, which represents a trivial braid of $n$ strands, and let $\sigma_1, \dots, \sigma_{n-1}$ be the Artin generators of this group. Each generator $\sigma_i$ represents the braid that transposes the $i$-th and $i+1$-th strands, while leaving the rest of the strands straight, and such that if the braid is seen as running from bottom to top, the transposition follows the right-hand rule direction.

Then we consider a factorization $b = \tau_1 \cdot \ldots \cdot \tau_k$ of $b$, with $\tau_i \in \{e, \sigma_1, \sigma_1^{-1}, \dots, \sigma_{n-1}, \sigma_{n-1}^{-1}\}$ for every $i$. Let us notice that we allow the redundant and possibly repeated presence of $e$ in the word $\tau_1 \cdot \ldots \cdot \tau_k$. We do not think of $b$ as an abstract braid but rather as a subspace of $\mathbb{R}^3$.

Let us denote the planes of $\mathbb{R}^3$ by writing $r$ and the equation of the plane as a subindex, and let $P$ and $Q$ denote the sets $\bigcup P_j$ and $\bigcup Q_j$ respectively. Let $D_0$ and $D_c$ be closed disks contained in $r_{z=0}$ and $r_{z=c}$ respectively, and such that $P$ is contained in the interior of $D_0$ and $Q$ in that of $D_c$. Moreover, let us choose these disks in such a way that $b$ is contained in a closed cylinder $C$, with bottom equal to $D_0$ and top equal to $D_c$, and satisfying that $[\partial C \backslash (D_0 \cup D_c)] \cap b = \emptyset$.

Let us assume that there is a set of planes $r_{z=z_1}, \ldots, r_{z=z_{k-1}}$ such that every $r_{z=z_i}$ intersects $b$ in the set of points $\{(j, 0, z_i)\}_{j=1}^{n}$, and such that the braid running from $r_{z=z_{i-1}}$ to $r_{z=z_i}$ is exactly $\tau_i$. This can be assumed without loss of generality by deforming $b$ inside $C$, and means that the braids $\tau_1, \ldots, \tau_k$ are disposed in strictly ascending order. Furthermore, if we define $c = k$, we can assume that $z_i = i$.

For $1 \leq j \leq n$ and $1 \leq i \leq k+1$ let us define $A_{j(i)} := (j, 0, i-1)$. We define also

$$\begin{aligned} D_i &:= C \cap r_{z=i-1}, \\ C_i &:= \{(x, y, z) \in C \mid i-1 \leq z < i\}. \end{aligned}$$

Then $C_i$ is a sub-cylinder of $C$ between $D_i$ and $D_{i+1}$ and $b \cap D_i = \{A_{j(i)}\}$. We can see the cylinder $C$ illustrated in Figure 1.3.

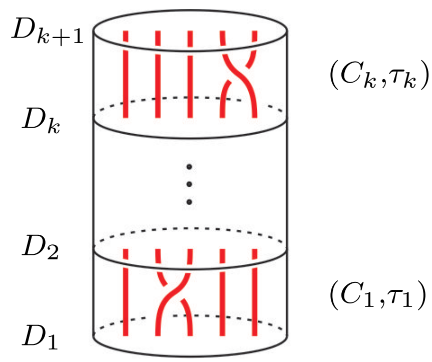


**Figure 1.3**

Let $\tau_i$ be a fixed arbitrary element of $\{\tau_1, \ldots, \tau_k\}$. Let us notice that $\tau_i \subset C_i$ and runs from $\{A_{j(i)}\}$ to $\{A_{j(i+1)}\}$. Let $h_1, \ldots, h_n$ be the strands of $\tau_i$, where $h_\ell$ is the strand starting at $A_{\ell(i)}$ and finishing at some element of $\{A_{j(i+1)}\}$.

If $\tau_i$ is the trivial braid we can assume, without loss of generality, that $h_1, \ldots, h_n$ are vertical line segments, and thus contained in $r_{y=0}$. Let us consider the set $(C_i \cap r_{y=0}) \diagdown \cup h_n$, which is a union of $n+1$ disjoint rectangular-shaped topological disks that are neither open nor closed. Let $\varsigma_0, \ldots, \varsigma_n$ be the closures of these disks. Let us observe that $\varsigma_0, \ldots, \varsigma_n$ split $C_i$ into two combinatorial $(n+2)$-gonal prisms, attached by $n+1$ rectangular faces. Since $C_i \diagdown (\partial C_i \cup \varsigma_0 \cup \cdots \cup \varsigma_n)$ is the union of two disjoint three-dimensional open balls, and $\partial C_i \cup \varsigma_0 \cup \cdots \cup \varsigma_n$ is a two-dimensional CW complex, of which $\tau_i$ is a subcomplex, then $\partial C_i \cup \varsigma_0 \cup \cdots \cup \varsigma_n$ provide us a CW decomposition for $(C_i, \tau_i)$.

**Definition 1.2.** We call $C_i$, endowed with this CW complex structure, a *double prism* for $\tau_i = e$.

We see the double prism illustrated in Figure 1.4 with names and orientations for each cell.

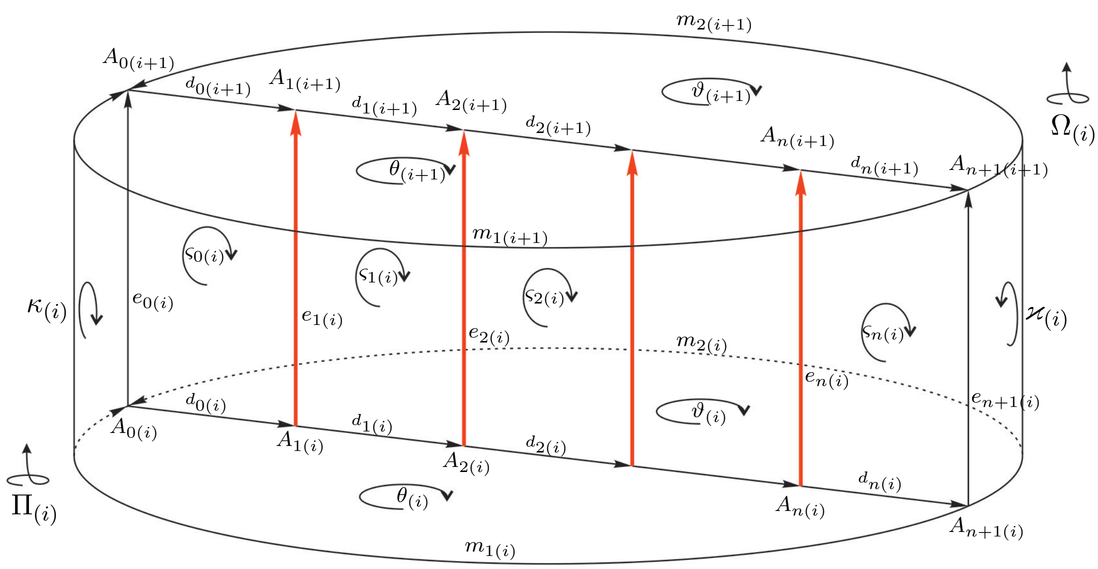


**Figure 1.4:** Here, $\Pi_{(i)}$ and $\Omega_{(i)}$ denote the interior of the prisms.

The boundaries of the cells, homologicaly speaking, are given below.

Dim. 1

$$
\begin{aligned}
\partial(e_{j(i)}) &= A_{j(i+1)} - A_{j(i)}\\
\partial(d_{j(i)}) &= A_{j+1(i)} - A_{j(i)}\\
\partial(m_{1(i)}) &= A_{0(i)} - A_{n+1(i)}\\
\partial(m_{2(i)}) &= A_{0(i)} - A_{n+1(i)}
\end{aligned}
$$

Dim. 2

$$
\begin{aligned}
\partial(\varsigma_{j(i)}) &= e_{j(i)} + d_{j(i+1)} - e_{j+1(i)} - d_{j(i)}\\
\partial(\kappa_{(i)}) &= m_{1(i)} + e_{0(i)} - m_{1(i+1)} - e_{n+1(i)}\\
\partial(\varkappa_{(i)}) &= m_{2(i)} - e_{n+1(i)} - m_{2(i+1)} + e_{0(i)}\\
\partial(\theta_{(i)}) &= m_{1(i)} + d_{0(i)} + \cdots + d_{n(i)}\\
\partial(\vartheta_{(i)}) &= -m_{2(i)} - d_{n(i)} - \cdots - d_{0(i)}
\end{aligned}
$$

Dim. 3

$$
\begin{aligned}
\partial(\Pi_{(i)}) &= \kappa_{(i)} - \theta_{(i)} + \theta_{(i+1)} - \varsigma_{0(i)} - \cdots - \varsigma_{n(i)}\\
\partial(\Omega_{(i)}) &= -\varkappa_{(i)} - \vartheta_{(i)} + \vartheta_{(i+1)} + \varsigma_{0(i)} + \cdots + \varsigma_{n(i)}
\end{aligned}
$$

We will examine now the case in which $\tau_i$ is an Artin generator. In this case every strand on $\{h_1,\dots,h_n\}$, except for two of them, connect some point of $\{A_{j(i)}\}$ with the point of $\{A_{j(i+1)}\}$ that has the same subindex, that is, the one that is directly above. These strands can be assumed to be vertical line segments contained in $r_{y=0}$. The remaining two strands, which we can assume are $h_1$ and $h_2$, suffer a transposition, with $h_1$ connecting $A_{1(i)}$ with $A_{2(i+1)}$, and $h_2$ connecting $A_{2(i)}$ with $A_{1(i+1)}$.

Let us take the points $\{A_{1(i)}, A_{2(i)}, A_{1(i+1)}, A_{2(i+1)}\}$ into consideration. These four points span a geometrical rectangle with boundary $R$ homeomorphic to $S^1$. Now imagine that $R$ is the equator of a convex three-dimensional topological ball that we call $\Phi$. This ball can be assumed convex and thin enough to ensure that $\Phi \setminus R \subset \mathring{C}_i$. Then, we can deform $h_1$ and $h_2$ to make them run along the boundary of $\Phi$, as it is illustrated in Figure 1.5.

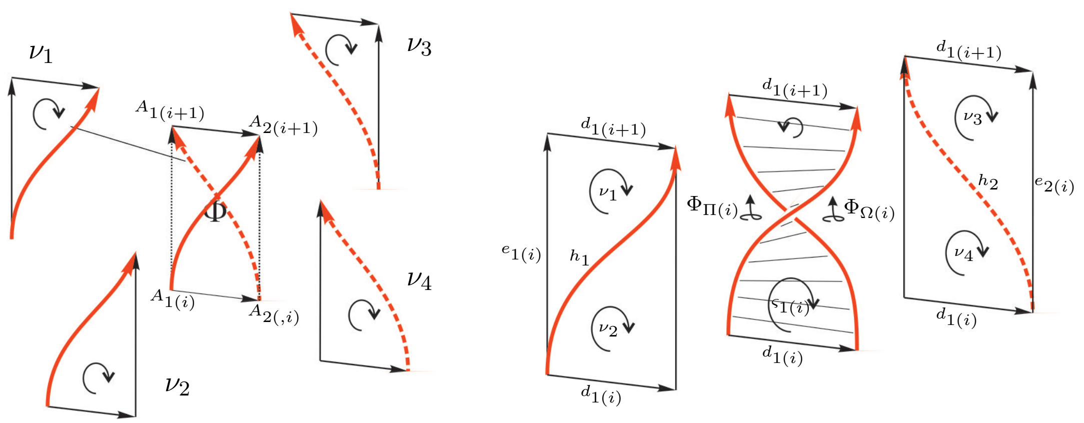


**Figure 1.5**

Let us consider the segments $\overline{A_{1(i)}A_{2(i)}}$ and $\overline{A_{1(i+1)}A_{2(i+1)}}$ along with the strands $h_1$ and $h_2$. The union of these four paths is homeomorphic to $S^1$. Let us consider a topological disk bounded by such union. We will call this disk $\varsigma_1$, and assume that its interior is contained in the interior of $\Phi$. Then $\varsigma_1$ is a combinatorial quadrilateral that ascends making a half twist. In addition, $h_1$, $h_2$ and $R$ split $\partial\Phi$ into four triangles, forming a combinatorial tetrahedron. We name these triangles as follows:

$$\begin{aligned}
\nu_1 &= A_{1(i)}A_{1(i+1)}A_{2(i+1)},\\
\nu_2 &= A_{1(i)}A_{2(i+1)}A_{2(i)},\\
\nu_3 &= A_{2(i)}A_{1(i+1)}A_{2(i+1)},\\
\nu_4 &= A_{1(i)}A_{1(i+1)}A_{2(i)}.
\end{aligned}$$

Let us observe also that $\varsigma_1$ splits $\Phi$ into two three-dimensional balls. We call $\Phi_\Pi$ (respectively $\Phi_\Omega$) the ball whose boundary contains $\nu_1$ (res. $\nu_3$). These objects are shown in Figure 1.5.

Finally, from the intersection $C_i \cap r_{y=0}$ let us subtract the set $\Phi \cup h_1 \cup \cdots \cup h_n$. The resulting set is a union of $n$ disjoint rectangular-shaped topological disks. Let $\varsigma_0, \varsigma_2, \ldots, \varsigma_n$ be the closures of these disks, ordered by crescent $x$ coordinates.

Let us observe that $C_i \setminus (\partial C_i \cup \varsigma_0 \cup \cdots \cup \varsigma_n \cup \nu_1 \cup \cdots \cup \nu_4)$ is the union of four disjoint three-dimensional open balls. Since $\partial C_i \cup \varsigma_0 \cup \cdots \cup \varsigma_n \cup \nu_1 \cup \cdots \cup \nu_4$ is a two-dimensional CW complex, of which $\tau_i$ is a subcomplex, it provides us a CW decomposition for $(C_i, \tau_i)$.

**Definition 1.3.** We call $C_i$ endowed with this CW complex structure a *double quasi-prism* for $\tau_i$.

For a general Artin generator $\sigma_q^{\pm 1}$ we assume that the twisted quadrilateral enclosed by $\Phi$ is $\varsigma_q$. Figure 1.6 illustrates a double quasi-prism, for a general right-handed twist $\sigma_q$, with names given to each cell. We use the same names for a left-handed twist (Let us notice that the decompositions of $\sigma_q$ and $\sigma_q^{-1}$ have the same set of cells and the same boundaries up to dimension two).

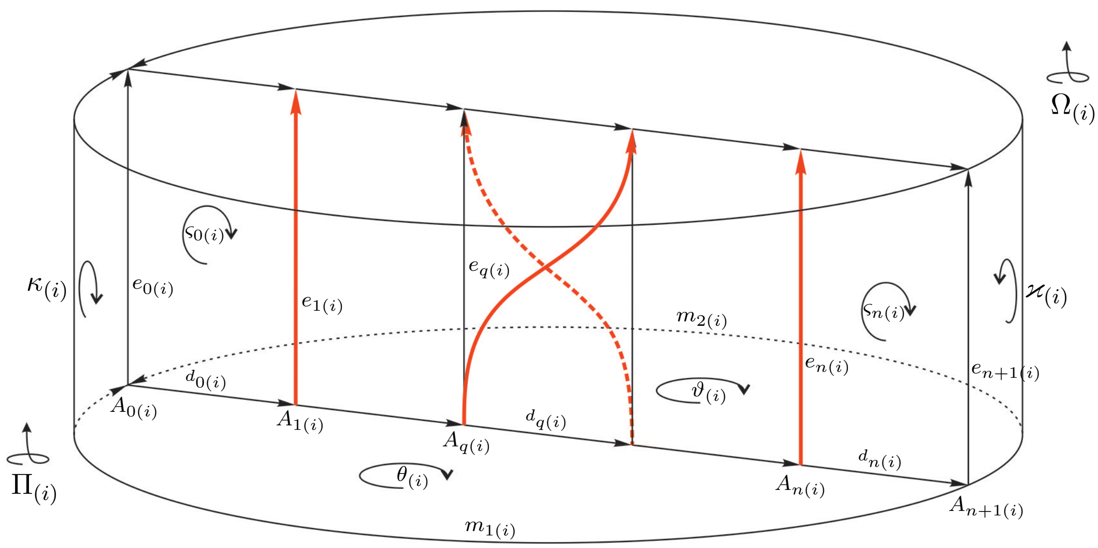


**Figure 1.6**

For a general generator $\sigma_q^s$, with $s = \pm 1$, the boundaries of the cells are given below, where the underlines mean that the underlined terms are to be omitted.

Dim. 1

$$
\begin{aligned}
\partial(e_{j(i)}) &= A_{j(i+1)} - A_{j(i)}\\
\partial(d_{j(i)}) &= A_{j+1(i)} - A_{j(i)}\\
\partial(m_{1(i)}) &= A_{0(i)} - A_{n+1(i)}\\
\partial(m_{2(i)}) &= A_{0(i)} - A_{n+1(i)}\\
\partial(h_{q(i)}) &= A_{q+1(i+1)} - A_{q(i)}\\
\partial(h_{q+1(i)}) &= A_{q(i+1)} - A_{q+1(i)}
\end{aligned}
$$

Dim. 2

$$
\begin{aligned}
\partial(\varsigma_{j(i)}) &= e_{j(i)} + d_{j(i+1)} - e_{j+1(i)} - d_{j(i)}, \quad j \neq q\\
\partial(\varsigma_{q(i)}) &= h_{q(i)} - d_{q(i+1)} - h_{q+1(i)} - d_{q(i)}\\
\partial(\nu_{1(i)}) &= d_{q(i+1)} - h_{q(i)} + e_{q(i)}\\
\partial(\nu_{2(i)}) &= -d_{q(i)} - e_{q+1(i)} + h_{q(i)}\\
\partial(\nu_{3(i)}) &= d_{q(i+1)} - e_{q+1(i)} + h_{q+1(i)}\\
\partial(\nu_{4(i)}) &= -d_{q(i)} + e_{q(i)} - h_{q+1(i)}\\
\partial(\kappa_{(i)}) &= m_{1(i)} + e_{0(i)} - m_{1(i+1)} - e_{n+1(i)}\\
\partial(\varkappa_{(i)}) &= m_{2(i)} - e_{n+1(i)} - m_{2(i+1)} + e_{0(i)}\\
\partial(\theta_{(i)}) &= m_{1(i)} + d_{0(i)} + \cdots + d_{n(i)}\\
\partial(\vartheta_{(i)}) &= -m_{2(i)} - d_{n(i)} - \cdots - d_{0(i)}
\end{aligned}
$$

Dim 3.

$$
\begin{aligned}
&\partial(\Pi_{(i)}) = \kappa_{(i)} - \theta_{(i)} + \theta_{(i+1)} - \varsigma_{0(i)} - \cdots - \underline{\varsigma_{q(i)}} - \cdots - \varsigma_{n(i)} - \nu_{2-s(i)} - \nu_{3-s(i)}\\
&\partial(\Omega_{(i)}) = -\varkappa_{(i)} - \vartheta_{(i)} + \vartheta_{(i+1)} + \varsigma_{0(i)} + \cdots + \underline{\varsigma_{q(i)}} + \cdots + \varsigma_{n(i)} + \nu_{2+s(i)} + \nu_{3+s(i)}\\
&\partial(\Phi_{\Pi(i)}) = s(\nu_{1(i)} - \nu_{4(i)} + \varsigma_{q(i)})\\
&\partial(\Phi_{\Omega(i)}) = s(\nu_{2(i)} - \nu_{3(i)} - \varsigma_{q(i)})
\end{aligned}
$$

If we endow each $(C_i, \tau_i)$ in $(C, b)$ with a double prism or a double quasi-prism structure we obtain a CW decomposition for $(C, b)$. Let us notice that this decomposition is a CW complex built only from a factorization $\tau_1 \cdot \ldots \cdot \tau_k$ of $b$.

**Definition 1.4.** We call $(C, b)$ endowed with this CW complex structure a *loom* for $b = \tau_1 \cdot \ldots \cdot \tau_k$.

**Theorem 1.2.** Let $b$ be a braid of $n$ strands and $\tau_1 \cdot \ldots \cdot \tau_k$ a factorization of $b$. A loom for $b = \tau_1 \cdot \ldots \cdot \tau_k$ is a well defined regular CW decomposition for $(C, b)$.

*Proof.* It follows from the construction that a loom for any factorization of any braid $b$ is a well defined regular CW complex. We need to see that, although looms arising from different factorizations of $b$ are combinatorially different, they yield the same underlying pair of spaces, which is $(C, b)$.

Let $\tau_1 \cdot \ldots \cdot \tau_k$ and $\tau'_1 \cdot \ldots \cdot \tau'_{k'}$ be two words presenting the same algebraic braid in Artin's presentation. That these words present the same topological braid $(C, b)$ is a basic fact of braid theory. That the underlying pair of spaces of a loom for either $\tau_1 \cdot \ldots \cdot \tau_k$ or $\tau'_1 \cdot \ldots \cdot \tau'_{k'}$ is $(C, b)$ is clear by construction. □

## 1.3 Decomposition of the Cone of a Three-Sphere Containing a Closed Braid

Let $b$ be a braid of $n$ strands as in the previous section. Let us deform $C$ in $\mathbb{R}^3$ and glue $D_1$ with $D_{k+1}$ to form a solid torus $C'$, in such a way that, for every $j$, $P_j$ is glued with $Q_j$. Now let us embed $C'$ in $S^3$. By this process $b$ is transformed into a closed braid or link that we call the closure of $b$ and denote by $\bar{b}$.

Let us consider $\bar{b}$ embedded in a three-dimensional sphere $S$ in this way. Then, the cone of $S$ is a four-dimensional ball containing the cone of $\bar{b}$, which is two-dimensional. Although the symbol $\vee$ is usually used for logical disjunction or the wedge sum of spaces, we will use it along this chapter to denote the cone of a space. Thus, let $\vee S$ be the cone of $S$ and $\vee\bar{b}$ the cone of $\bar{b}$ inside of $\vee S$. Our purpose now is to construct a CW the pair $(\vee S, \vee\bar{b})$. We do this by constructing $\vee S$ through a series of steps.

We start the construction from a cylinder containing a braid. Let $C$ be a loom for $b$. The plane $r_{y=0}$ intersects $\partial C$ at the union of four line segments, two of which are vertical. These segments will be called $a_1$ and $a_2$, being $a_1$ the closest to the $z$ axis and $a_2$ the farthest. Let us recall that the CW complex structure of $C$ is dependent on the factorization $b = \tau_1 \cdot \ldots \cdot \tau_k$, where $\tau_i \in \{e, \sigma_1, \sigma_1^{-1}, \ldots, \sigma_{n-1}, \sigma_{n-1}^{-1}\}$ for each $i$. Let us assume $\tau_1 = e$, and construct the CW complex accordingly. This apparently superfluous requirement has the purpose to facilitate later constructions.

As a second step, we identify the disks $D_1$ and $D_{k+1}$, according to a homeomorphism constant on the $x$ and $y$ variables. By doing this, we transform $C$ into a solid torus, that we call $T_1$, and $b$ into the closed braid $\bar{b}$. Also, the sets resulting from $C_1, \ldots, C_k$ and $D_1, \ldots, D_k$ will preserve these names, with the disk resulting from the identification of $D_1$ and $D_{k+1}$ being called $D_1$. It is clear that $T_1$ has a CW complex structure directly inherited from $C$. We call $T_1$ endowed with this structure a *closed loom* for $b$.

The third step will be to glue $T_1$ to another solid torus to complete a sphere. Let $T_2$ be a solid torus, let $\mu$ and $\lambda$ be two disjoint meridian disks of $T_2$, and let $l$ be a longitude of $T_2$ (homologous to the generator of $H_1(T_2)$) such that $\mu \cap l$ and $\lambda \cap l$ are single points. Now we glue $T_1$ and $T_2$ by their boundaries according to a homeomorphism $\varphi : \partial T_1 \longrightarrow \partial T_2$ such that

$$\varphi(a_1) = \partial\mu, \;\; \varphi(a_2) = \partial\lambda, \;\; \varphi(\partial D_1) = l.$$

We denote the three-dimensional sphere $T_1 \cup_\varphi T_2$ by $S$. It can be readily seen that $S$ has a CW complex structure induced by those of $T_1$ and $T_2$. The zero and one-dimensional cells of this structure are those of $T_1$; the two-dimensional cells are those of $T_1$ with the addition of $\mu$ and $\lambda$; and the three-dimensional cells are those of $T_1$ with the addition of the two balls composing $T_2 \diagdown (\partial T_2 \cup \mu \cup \lambda)$.

The last step is to endow $\vee S$ with the CW complex structure conically induced by the structure of $S$. Let us notice that the resulting CW complex is built only from a factorization $\tau_1 \cdot \ldots \cdot \tau_k$ of $b$.

**Definition 1.5.** We call $\vee S$ endowed with this CW complex structure a *marble* for $b = \tau_1 \cdot \ldots \cdot \tau_k$.

This complex is shown in Figure 1.7.

**Theorem 1.3.** Let $b$ be a braid of $n$ strands, $\tau_1 \cdot \ldots \cdot \tau_k$ a factorization of $b$, and $\bar{b}$ its closure. A marble for $b = \tau_1 \cdot \ldots \cdot \tau_k$ is a well defined regular CW decomposition for $(\vee S, \vee\bar{b})$.

*Proof.* It is clear by construction and Theorem 1.2. □

The following observation implies that the topology of the underlying pair of the CW complex is not only independent of the factorization of $b$, but independent on this factorization up to conjugation.

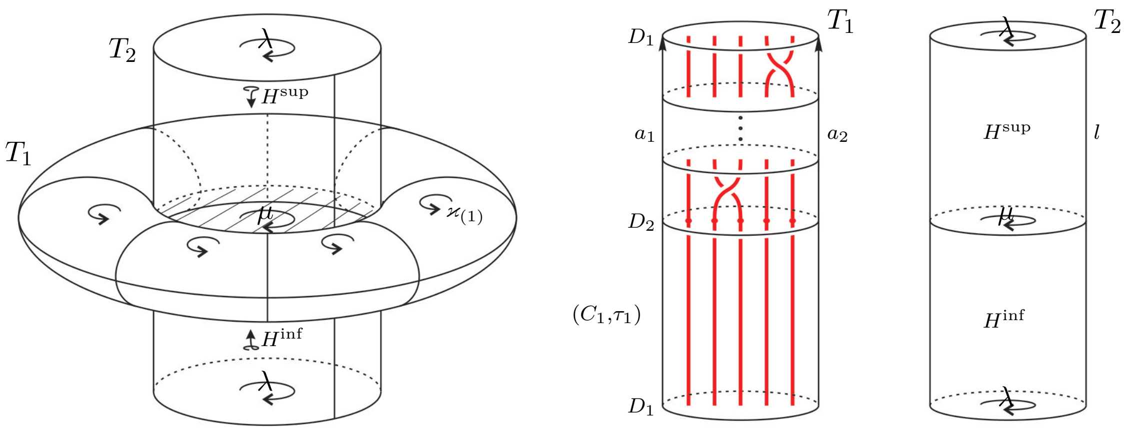


**Figure 1.7**

**Observation 1.4.** Let $a, b \in \mathcal{B}_n$. By Markov's Theorem ([8, Theorem 2.3]), if $a$ and $b$ are conjugated braids in $\mathcal{B}_n$, then $\bar{a} = \bar{b}$. Therefore, the marbles for any factorizations of two conjugated braids $a$ and $b$ are decompositions of the pairs $(\vee S, \vee\bar{a})$ and $(\vee S, \vee\bar{b})$, and thus are topologically equivalent (though not combinatorially equivalent).

Let us consider now the two balls that compose $T_2 \diagdown (\partial T_2 \cup \mu \cup \lambda)$. One of these balls has its boundary attached to points of $\partial T_1$ coming from points of $\mathbb{R}^3$ with positive $y$ coordinate. A similar statement is true for the other ball, but for points with negative $y$ coordinate. These two balls will be called respectively the *superior* and *inferior caps* of $\vee S$, and will be denoted by $H^{\text{sup}}$ and $H^{\text{inf}}$.

We will describe now the cells composing $\vee S$ and its boundaries. Let $AA$ denote the apex of $\vee S$ (The double $A$ is only a name and has the purpose to distinguish this vertex from other vertices that are to be called $A$). Let us notice that the cells of $\vee S$ are divided into three (disjoint) sets: The set $\text{Bnd}(\vee S)$ of all the cells of $S$; the set $\text{Con}(\vee S)$ of all the conical cells produced by taking the cone of each cell of $S$; and the singleton $\{AA\}$.

We begin with the set $\text{Bnd}(\vee S)$, which is composed by $\mu$, $\lambda$, $H^{\text{sup}}$, $H^{\text{inf}}$ and all the cells of $T_1$. The torus $T_1$ is in turn composed by a series of prisms and quasi-prisms $C_1, \ldots, C_k$. For each $C_i$ we have the cells and boundaries already described in the previous section, only that the identification of $D_1$ and $D_{k+1}$ implies that the subindex $i$ now has to be taken modulus $k$. Then, it only remains to provide the boundaries of $\mu$, $\lambda$, $H^{\text{sup}}$ and $H^{\text{inf}}$, which are given below.

$$\begin{aligned}
\partial(\mu) &= e_{0(1)} + \cdots + e_{0(k)} \\
\partial(\lambda) &= e_{n+1(1)} + \cdots + e_{n+1(k)} \\
\partial(H^{\text{sup}}) &= \mu - \lambda - \varkappa_{(1)} - \cdots - \varkappa_{(k)} \\
\partial(H^{\text{inf}}) &= \mu - \lambda - \kappa_{(1)} - \cdots - \kappa_{(k)}
\end{aligned}$$

Now we consider the set $\mathrm{Con}(\vee S)$. For any cell $\rho$ of $S$, let $\vee\rho$ denote the cone of $\rho$. Moreover, for any chain $c = a_1\rho_1 + \cdots + a_l\rho_l$ of cells of a given dimension of $S$, let $\vee c$ denote the chain $a_1 \vee \rho_1 + \cdots + a_l \vee \rho_l$. We give each cell $\vee\rho$ the orientation resulting by adding to the orientation of $\rho$ the vertex $AA$. Then, the boundaries of the conical cells are given by

$$\partial(\vee\rho) = AA - \rho$$

if $\dim(\rho) = 0$, and by

$$\partial(\vee\rho) = (-1)^{\dim(\rho)}(-\rho) + \vee(\partial(\rho))$$

if $\dim(\rho) > 0$.

## 1.4 Local Decomposition for the Zero Set of a Function in $\mathbb{C}\{x, y\}$

Let us consider again $f$ and $\Omega$ as in the first section. We are now interested in finding a CW decomposition of a small neighborhood around a point $p$ of $\Omega$.

By a translation, we may assume that $p$ is the origin. Now we consider $f$ as an element of $\mathbb{C}\{x, y\}$. In fact, since we are only going to examine $f$ locally, what we discuss in this section is valid for any function of $\mathbb{C}\{x, y\}$ sending zero to zero, not necessarily a polynomial. Therefore, in this section, we consider $f$ as an arbitrary element of $\mathbb{C}\{x, y\}$ such that $f(0, 0) = 0$.

Let $D^\varepsilon$ and $D^\eta$ be disks in $\mathbb{C}$ centered at the origin with radii $\varepsilon$ and $\eta$ respectively, and such that $f$ is convergent in the polydisc $D^\varepsilon \times D^\eta$. The radii $\varepsilon$ and $\eta$ are to be shrunken if necessary. Also, for any function of $\mathbb{C}\{x, y\}$ convergent on $D^\varepsilon \times D^\eta$, let $V(\cdot)$ denote the zero set of that function in $D^\varepsilon \times D^\eta$.

Let us recall that an element $w$ of $\mathbb{C}\{x, y\}$ is called a *Weierstrass polynomial* (with respect to $y$) if it is of the form

$$w(x, y) = y^d + a_1(x)y^{d-1} + \cdots + a_{d-1}(x)y + a_d(x),$$

where, for every $i$, $a_i(x) \in \mathbb{C}\{x\}$ and $a_i(0) = 0$.

As in the first section, by a change of variable, we may assume that $f(0, y)$ is not identically zero. Then, by the Weierstrass Preparation Theorem, there exists a unit $u \in \mathbb{C}\{x, y\}$ and a Wierstrass polynomial $w \in \mathbb{C}\{x\}[y]$ such that, inside a certain neighborhood of $(0, 0)$,

$$f(x, y) = u(x, y)w(x, y).$$

By shrinking $\varepsilon$ and $\eta$, we may assume that the former equality holds in $D^\varepsilon \times D^\eta$. Furthermore, the fact that $u$ is a unit in $\mathbb{C}\{x, y\}$ means that $u(0, 0) \neq 0$, so choosing $\varepsilon$ and $\eta$ small enough we can ensure that $V(f) = V(w)$. This means that close to the origin we can work with $w$ instead of $f$ for geometrical reasonings.

On the other hand, we know that the factorization of $w$ into irreducible factors in $\mathbb{C}\{x\}[y]$ (and in $\mathbb{C}\{x,y\}$) is of the form

$$w(x,y) = w_1^{k_1}(x,y)\cdots w_l^{k_l}(x,y)$$

where $w_1,\ldots,w_l$ are Weierstrass polynomials. Then,

$$V(w) = V(w_1)\cup\cdots\cup V(w_l).$$

The set $V(w)$ is called a *local analytic curve in the neighborhood of* $(0,0)$, while each $V(w_r)$ is called an *irreducible local analytic curve component of* $V(w)$.

By the Local Normalization Theorem for Algebraic Curves, each $V(w_r)$ is a disk centered at the origin, and the boundary of $V(w)$ is a link in $\partial(D^\varepsilon\times D^\eta)$. This local study of curves by means of their Weierstrass polynomials is classical and is exposed with detail in [14].

On the other hand, we may assume that $w$ has not multiple factors (i.e. $k_1=\cdots=k_l=1$) because, if it had them, they could be removed without altering $V(w)$. Then we have the following.

**Claim 1.5.** *For every $i\neq j$, $V(w_i)\cap V(w_j)=(0,0)$.*

*Proof.* We reason in a similar way as in Claim 1.1. By taking $A=\mathbb{C}\{x\}$, we have that $disc_y(w)$ is a series belonging to $\mathbb{C}\{x\}$, that we will call $D(x)$. Since $W$ has not multiple factors, $D(x)$ is not identically zero.

Let us observe that $D(x)$ can be factorized as $D(x)=x^mQ(x)$, with $m\geq 0$ and $Q(x)\in\mathbb{C}\{x\}$ satisfying that $Q(0)\neq 0$. Let us reduce $\varepsilon$ enough so we can ensure that $Q(x)\neq 0$ for every $x\in D^\varepsilon$. Therefore, $D(x)\neq 0$ for every $x\in D^\varepsilon\setminus\{(0,0)\}$.

From here it follows that $w$ has not multiple roots in $D^\varepsilon\setminus\{(0,0)\}$, and therefore $V(w_i)$ and $V(w_j)$ do not intersect in a point other than $(0,0)$. $\square$

Now, let $\check{x}$ be a fixed point of $\partial D^\varepsilon$, and let $\check{y}_1,\ldots,\check{y}_n$ be the roots of $w(\check{x},y)$. Then the points of $V(w)$ with first coordinate equal to $\check{x}$ are $(\check{x},\check{y}_1),\ldots,(\check{x},\check{y}_n)$. If we move $\check{x}$ over $\partial D^\varepsilon$ all the way around $D^\varepsilon$ completing the circumference, the points $(\check{x},\check{y}_1),\ldots,(\check{x},\check{y}_n)$ will travel accordingly inside $\partial D^\varepsilon\times\mathbb{C}$ completing a closed braid. Let $b$ be a braid such that its closing produces this closed braid. Then $b$ is a local braid of $w$ along $\partial D^\varepsilon$ with a given orientation.

Besides, $\bar{b}=\bigcup\partial V(w_r)$ is contained in $\partial(D^\varepsilon\times D^\eta)$. Since the $V(w_r)$ are disjoint disks except by the origin, $(D^\varepsilon\times D^\eta, V(w))$ is the cone of $(\partial(D^\varepsilon\times D^\eta),\bar{b})$ with apex at he origin. Hence, we can then apply Theorem 1.3 and give $(D^\varepsilon\times D^\eta, V(w))$ a CW complex structure. Since two different SCP at $\check{x}$ yield conjugated braids, Observation 1.4 ensures that the pair $(\vee S,\vee\bar{b})$ is topologically equivalent to $(D^\varepsilon\times D^\eta, V(w))$, regardless of the choice of $b$.

Furthermore, since $b\subset\partial D^\varepsilon\times D^\eta$, we can take $T_1=\partial D^\varepsilon\times D^\eta$ and $T_2=D^\varepsilon\times\partial D^\eta$.

## 1.5 Decomposition for a Local Braid

Let us consider again $f$ and $\Omega$ as in the first section. Let $D^r$ and $D^r_p$ denote closed disks in $\mathbb{C}$, centered at the origin and at point $p$ respectively, of radius $r$, and to be shrunken if necessary. Let $(\hat{x}, \hat{y}) \in \Omega$. If we wanted a purely local description of the embedding of $\Omega$ in $\mathbb{C}^2$, we could take a small polydisc $D := D^{\varepsilon}_{\hat{x}} \times D^{\eta}_{\hat{y}}$ around $(\hat{x}, \hat{y})$ and then apply Theorem 1.3 just as in Section 1.4 to obtain a CW decomposition of $(D, \Omega \cap D)$. In this section, however, we aim a little more.

We want our polydisc to contain not only the points of $\Omega$ near $(\hat{x}, \hat{y})$, but all the points of $\Omega$ near any point $(\hat{x}, y) \in \Omega$. Thus, intuitively speaking, our polydisc $D$ will be the product of a small disk $D^{\varepsilon}_{\hat{x}}$ and a big disk $D^{\eta}_{\hat{y}}$, in such a way that $D = D^{\varepsilon}_{\hat{x}} \times D^{\eta}_{\hat{y}}$ contains all the points in $\Omega \cap (D^{\varepsilon}_{\hat{x}} \times \mathbb{C})$. Moreover, since we are interested in describing the topology of the embedding of the curve into $(D^{\varepsilon}_{\hat{x}} \times \mathbb{C})$, we will not demand $D$ to be a polydisc, but only a four-dimensional ball satisfying that $\Omega \cap D = \Omega \cap (D^{\varepsilon}_{\hat{x}} \times \mathbb{C})$. It can be seen that any two four-dimensional balls (and in particular polydiscs) satisfying this and an additional condition on the boundary are ambient isotopic in $\mathbb{C}^2$, by an isotopy leaving $\Omega$ constant. We will construct now a CW decomposition for $(D, \Omega \cap D)$.

Now we consider $f$ as an element of $\mathbb{C}\{x - \hat{x}\}[y]$. In fact, since we are only going to examine $f$ in values of $x$ close to $\hat{x}$, we will work in the wider context of a function $f \in \mathbb{C}\{x - \hat{x}\}[y]$. Therefore, in this section, we consider $f$ as an arbitrary element of $\mathbb{C}\{x - \hat{x}\}[y]$.

Let $\hat{x}$ be a point in $\mathbb{C}$. As in the first section, we can assume $f \in \mathbb{C}\{x - \hat{x}\}[y]$ is of the form

$$f(x, y) = y^n + a_1 y^{n-1} + \cdots + a_{n-1} y + a_n,$$

where $a_i \in \mathbb{C}\{x - \hat{x}\}$ for every $i$. Let $D^{\varepsilon}_{\hat{x}}$ be a disk where every $a_i$ is convergent. Then $f$ will be considered as a function from $D^{\varepsilon}_{\hat{x}} \times \mathbb{C}$ into $\mathbb{C}$. We denote the zero set of $f$ by $V(f)$ as usual. Finally, let $\Delta$ be defined by

$$\Delta = \{x \in D^{\varepsilon}_{\hat{x}} \mid f(x, y) \text{ has multiple roots}\}.$$

We know that $\Delta$ is either empty or equal to $\{\hat{x}\}$, being the latter the interesting case.

**Claim 1.6.** *Either $\Delta = \emptyset$ or $\Delta = \{\hat{x}\}$.*

*Proof.* It can be proved by the same arguments used in Claims 1.1 and 1.5.

Let $p_1 = (\hat{x}, y_1), \dots, p_l = (\hat{x}, y_l)$ be the points of $V_{x=\hat{x}}$, with $l \leq n$. For $1 \leq r \leq l$, let $\Phi_r$ be the local analytic curve of $f$ in the neighborhood of $p_r$. For $\varepsilon$ small enough, we can assume that

$$V(f) \cap (D^{\varepsilon}_{\hat{x}} \times \mathbb{C}) = \Phi_1 \cup \cdots \cup \Phi_l.$$

We already know that each $\Phi_r$ is a union of topological disks identified by their centers. Besides, these disks are, except by their common centers, disjoint.

For $1 \leq r \leq l$, let $n_r$ denote the number of points in which the vertical line $L_{x=c}$ intersects $\Phi_r$, for $c \in D^{\varepsilon}_{\hat{x}} \setminus \{\hat{x}\}$; this is, the degree of the corresponding Weierstrass polynomial around $p_r$. Let us notice that if, for a certain $r$, the number $n_r$ is equal to 1, then $\Phi_r$ is a single disk transversal to $L_{x=\hat{x}}$. Also, since $f$ does not have multiple roots in $D^{\varepsilon}_{\hat{x}} \setminus \{\hat{x}\}$, the sets $\Phi_1, \dots, \Phi_l$ are pairwise disjoint.

We summarize these statements in the following lemma.

**Lemma 1.7.** *For $1 \leq r \leq l$, the set $\Phi_r$ is a disk if $n_r = 1$, and a union of at most $n_r$ independent topological disks with identified centers if $n_r > 1$. Also, for $j \neq r$, $\Phi_r \cap \Phi_j = \emptyset$.*

Now, let the loop $\gamma$ be defined by

$$\gamma(t) = \hat{x} + \varepsilon e^{2\pi i t},\ t \in I.$$

Let $a = \hat{x} + \varepsilon$, and let $\beta$ be the local braid of $f$ around $\hat{x}$ taken along $\gamma$, and defined according to an SCP $\rho = \{\rho_i\}_{i=1}^{n-1}$ in $L_{x=a}$.

Let us define the boundary $\partial\Phi_r$ of $\Phi_r$ as the union of the boundaries of the disks forming it. Then, for $1 \leq r \leq l$, the link $\partial\Phi_r$ is a union of components of $\bar{\beta}$, and

$$\bar{\beta} = \partial\Phi_1 \cup \cdots \cup \partial\Phi_l.$$

From here it follows that there is a sub-braid $\beta_{(r)}$ of $\beta$, defined according to $\rho$, such that its closure $\bar{\beta}_{(r)}$ is equal to $\partial\Phi_r$. This braid $\beta_{(r)}$ is the local braid along $\gamma$ of the Weierstrass polynomial around $p_r$ and, if we make $\varepsilon$ tend to 0, then $\beta_{(r)}$ tends to $p_r$. Thus, $\beta$ can be decomposed into the $l$ sub-braids $\beta_{(1)}, \dots, \beta_{(l)}$.

Let us consider the SCP $\rho$ for a moment. If, for each $r$, the curves of $\rho$ on $V_{x=a}$ join consecutively the points of $V_{x=a}$ corresponding to the strands $\beta_{(r)}$, we say that $\rho$ *separates* $\beta$ *into* $\beta_{(1)}, \dots, \beta_{(l)}$, that it is *a separating SCP for* $\beta$ or, by short, that it is an SSCP for $\beta$.

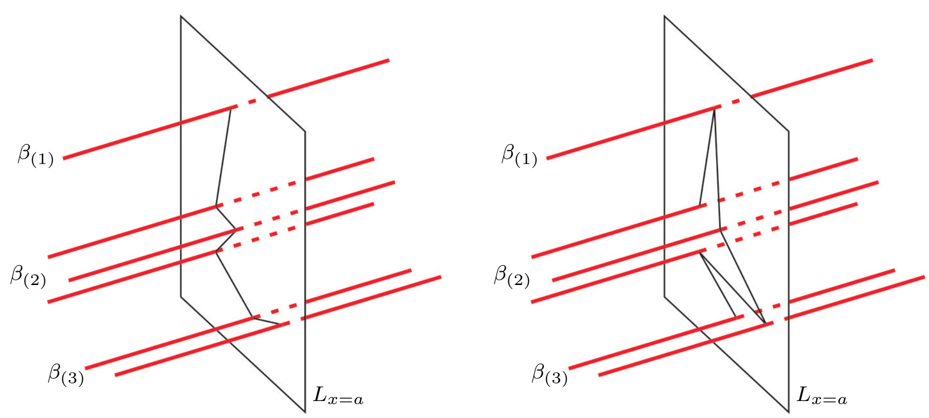


**Figure 1.8:** Here we see two different SCP at the same $a$. The one on the left separates $\beta$, while the one on the right does not.

**Observation 1.8.** Given $f$, $\hat{x}$, $\gamma$, and $\beta$ as before, for a small enough $\varepsilon$, the SCP at $\hat{x}+\varepsilon$ that joins through straight segments the points of $V_{\hat{x}+\varepsilon}$ in the lexicographical order of $\mathbb{C}$ is an SSCP for $\beta$.

It is important to notice that, as algebraic braids, $\beta_{(1)},\dots,\beta_{(l)}$ are also defined upon $\rho$. If $\rho$ is separating, we can consider each $\beta_{(r)}$ as defined upon an SCP $\rho_{(r)}$, consisting on the paths on $\rho$ joining the points of $V_{x=a}$ corresponding to $\beta_{(r)}$. The order that $\rho$ induces on the $\rho_{(r)}$ is also the order that $\beta$ induces on the $\beta_{(r)}$. Therefore, $\rho$ defines an ordered set $\{\rho_1,\dots,\rho_l\}$ of SCP, which in turn defines the ordered set of sub-braids $\{\beta_{(1)},\dots,\beta_{(l)}\}$. The following is a trivial but important observation.

**Observation 1.9.** If the algebraic braids $\beta,\beta_{(1)},\dots,\beta_{(l)}$ are defined upon an SSCP for $\beta$, then $\beta$ determines, and is determined, by the set $\{\beta_{(1)},\dots,\beta_{(l)}\}$ and a total order on this set.

From now on, we demand that the SCP $\rho$ defining $\beta$ is separating and that $\beta_{(1)},\dots,\beta_{(l)}$ are numbered in the order induced by $\rho$.

Now we are in a position to provide the decomposition of $(D,\Omega\cap D)$, as we proposed at the beginning of the section. We will do so by constructing the ball $D$ from smaller parts.

For $1\le r\le l$ let $H_r$ be a Milnor polydisc around $(\hat{x},y_r)$. Without loss of generality, by taking $\varepsilon$ small enough, we may assume that, for every $r$, $H_r$ is the product of $D^{\varepsilon}_{\hat{x}}$ with some disk $D_{y_r}$ in the $y$ variable. We may also assume that $H_1,\dots,H_l$ are pairwise disjoint. Let us observe also that

$$\begin{aligned} H_r\cap V(f) &= \Phi_r, \\ \partial H_r\cap V(f) &= \beta_r. \end{aligned}$$

Then, by the conical structure of the Milnor polydisc, we are in a condition to apply Theorem 1.3 and Observation 1.4 to each $H_r$ as in Section 1.4. Thus, we endow each $H_r$ with the CW complex structure referred to in the theorem, turning $H_r$ into a marble associated with some factorization of $\beta_{(r)}$.

In fact, we could define each $H_r$ more generally as a Milnor ball, and since a Milnor ball also has a conical structure, we could apply Theorem 1.3 in the same way. However, to define $H_r$ as a polydisc aids to the imagination because, on each $H_r$, the two solid tori $T_1$ and $T_2$ of the marble structure can be chosen to be the the two solid tori $\partial D^{\varepsilon}_{\hat{x}}\times D_{y_r}$ and $D^{\varepsilon}_{\hat{x}}\times\partial D_{y_r}$ naturally produced by the product structure of the polydisc.

Now we are going to glue all the marbles $H_1,\dots,H_l$ in a convenient way. Let us observe that, if we have two disjoint balls in $\mathbb{R}^3$, it is possible to deform them in such a way that, after the deformation, they intersect on a disk. Similarly, it is possible to deform two balls in $\mathbb{R}^4$, or in $\mathbb{C}^2$, in such a way that they intersect on a three-dimensional ball. In this way, we are going to deform $H_2$ in order that $H_1$ and $H_2$ intersect on a three-dimensional ball. Then we deform $H_3$ in order that $H_2$ and $H_3$ intersect on a three-dimensional ball, and so on, until we deform $H_l$ in order that $H_l$ and $H_{l-1}$ intersect in the same way.

It is obvious that, by doing these deformations, each $H_r \neq H_1$ ceases to be a geometrical polydisc. However, the marble structure is preserved, though deformed. It is clear also that at the end of the process the union of all the $H_r$ is a four-dimensional ball in $\mathbb{C}^2$. In order for this ball to have a CW complex structure, and intersects the curve in a nice subcomplex, these deformations must be done carefully. We are going to show now the exact way in which these deformations are to be done.

Since we allow the redundant presence of the identity in the factorization of any $\beta_{(r)}$, we may assume that the factorizations of all the braids $\beta_{(r)}$ have the same length $k$.

On the other hand, let us recall that any marble possesses a family of disks denoted by $D_i$, and a family of cylinders denoted by $C_i$, which are subcomplexes of its marble structure and were defined in the previous section. For each $r$, let $D_{(1,r)}, \ldots, D_{(k,r)}$ and $C_{(1,r)}, \ldots, C_{(k,r)}$ denote the disks $D_i$ and the cylinders $C_i$ of $H_r$. Let us denote also the superior and inferior caps of $H_r$ by $H^{\sup}_{(-,r)}$, and $H^{\inf}_{(-,r)}$. The subindex "$-$" has no true meaning here, and its use will be explained later.

We proceed to glue $H_1$ and $H_2$ as follows. Let us deform $H_2$ inside of $D^{\varepsilon}_{\hat{x}} \times \mathbb{C}$ (by an isotopy of its inclusion map), and leaving $\Phi_2$ fixed, in order that $\partial H_1$ and $\partial H_2$ intersect in a three-dimensional ball. We demand this deformation to be done without generating intersections with $H_3, \ldots, H_l$. This is possible because $p_1$ and $p_2$ can be joined by a path inside of $L_{x=\hat{x}}$ that does not intersect $H_3 \cup \cdots \cup H_l$. Then, the deformation can be done in the interior of a regular neighborhood of this path that does not intersect $H_3 \cup \cdots \cup H_l$. In other words, the deformation can follow this path, while keeping $H_2$ thin enough to avoid intersecting $H_3 \cup \cdots \cup H_l$. Therefore, since $\Phi_r \subset H_r$ for every $r \geq 3$, this deformation does not generate new intersections with $V(f)$ either.

The deformation can be done, moreover, in such a way that the following conditions hold.

- $(\partial H_1 \cap \partial H_2) = H_1 \cap H_2 = H^{\sup}_{(-,1)} = H^{\inf}_{(-,2)}$.
- The orientations of $H_1 \cap H_2$ inherited from $H_1$ and $H_2$ coincide.
- For every $0 \leq i \leq k$, the boundaries of $C_{(i,1)}$ and $C_{(i,2)}$ coincide in the intersection of $H_1$ and $H_2$, i.e. that $\partial C_{(i,1)} \cap H^{\sup}_{(-,1)} = \partial C_{(i,2)} \cap H^{\inf}_{(-,2)}$.

It is a trivial observation that this gluing is cellular. The resulting situation is illustrated in Figure 1.9.

Following the same procedure, we can deform $H_3$ in order that $H_2 \cap H_3 = H^{\sup}_{(-,2)} = H^{\inf}_{(-,3)}$, and that the same coincidence conditions hold. We continue this gluing process inductively until we reach $H_l$. We define then

$$T := H_1 \cup \cdots \cup H_l.$$

It follows from the construction that this set is a closed ball and, since we have glued all the $H_r$ cellularly, it has a CW complex structure trivially inherited from the $H_r$. Let us notice that the resulting CW complex is built only from the ordered set of sub-braids $\{\beta_{(1)}, \ldots, \beta_{(l)}\}$ and their factorizations.

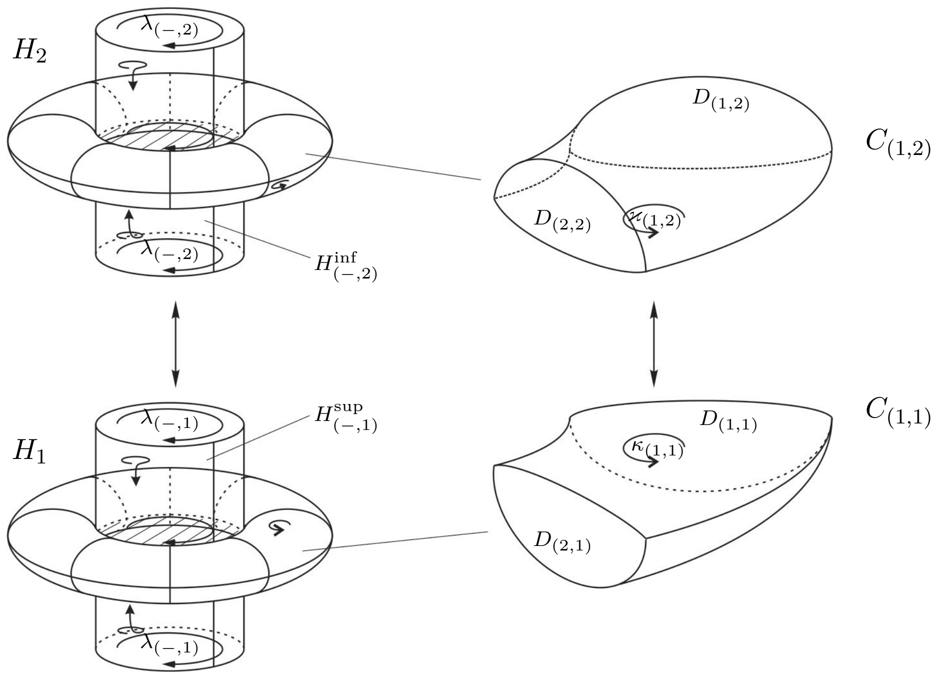


**Figure 1.9**

**Definition 1.6.** We call $T$ endowed with this CW complex structure a *tower* for $\hat{x}$.

The following theorem has already been proved along the construction.

**Theorem 1.10.** Let $a$, $\gamma$, and $\beta$ be as defined in this section. Let $\rho$ be an SCP at $a$ that separates $\beta$, defining the ordered set of sub-braids $\{\beta_{(1)}, \ldots, \beta_{(l)}\}$. A tower for $\hat{x}$, built upon the ordered set of sub-braids $\{\beta_{(1)}, \ldots, \beta_{(l)}\}$, and their factorizations, is a well defined regular CW decomposition for $(T, T \cap V(f))$, where $T$ is constructed as before.

Here, $T$ is by definition the underlying space of the tower. We will prove now a stronger version of this theorem (Theorem 1.13). Though it is not strictly necessary to our construction, this strongest version will allow us to consider $T$ as an arbitrary ball satisfying certain properties. In particular, it will allow us to consider $T$ as a polydisc of the form $D^{\varepsilon}_{\hat{x}} \times D$, for a large enough disk $D$ in $y$, as we posed in the beginning of this section. Also, for the rest of the section, we may assume we work in the smooth category.

Let us observe that $T$ satisfies the two following equalities:

$$\begin{aligned} \text{I. } T \cap V(f) &= (D^{\varepsilon}_{\hat{x}} \times \mathbb{C}) \cap V(f) = \Phi_1 \cup \cdots \cup \Phi_l. \\ \text{II. } \partial T \cap V(f) &= (\gamma \times \mathbb{C}) \cap V(f) = \bar{\beta}. \end{aligned}$$

The following lemma states that $T$ is in fact generic in regard to these equalities.

**Lemma 1.11.** *Let $U \subset D^{\varepsilon}_{\hat{x}} \times \mathbb{C}$ be homeomorphic to a closed ball and such that*

$U \cap V(f) = (D^{\varepsilon}_{\hat{x}} \times \mathbb{C}) \cap V(f)$ *and*
$\partial U \cap V(f) = \beta$.

*Then there exists an isotopy from $T$ into $U$ constant on $V(f)$.*

Before continuing, let us fix the following notation. Let $A$ and $B$ be topological spaces and $g, h : A \longrightarrow B$ homeomorphisms to their images. Then, to denote an isotopy $H : A \times I \longrightarrow B$, such that $H_0 = g$ and $H_1 = h$, we write $H : g \longrightarrow h$. Also, let $1_A$ denote the identity map on $A$. In order to prove Lemma 1.11 we need to show the following auxiliary fact first.

**Lemma 1.12.** *Let $h : T \longrightarrow U$ be an orientation-preserving homeomorphism. Then, there exists an isotopy of the identity $H : U \times I \longrightarrow U$ that, for every $r$, sends $\Phi_r$ into $h(\Phi_r)$.*

*Proof.* Let $B_1, \ldots, B_l$ be four-dimensional balls in $U$ such that, for every $r$, $\Phi_r \subset B_r$. Let us observe that each $B_r$ can be deformed (while leaving $\Phi_r$ constant) into a standard polydisc $D^{\varepsilon}_{\hat{x}} \times D^{\eta}_{y_r}$, for certain $y_r$ and $\eta$. Due to this, each $B_r$ has a product structure inherited from the polydisc, and can in fact be thought as the polydisc. For each $r$, let $T_r$ be the solid torus in $B_r$ corresponding to $\partial D^{\varepsilon}_{\hat{x}} \times D^{\eta}_{y_r}$. Then, for each $r$,

$$\beta_r \subset T_r \subset \partial B_r \subset \partial U.$$

By definition, $T_1, \ldots, T_l$ are disjoint and unknotted in $\partial U$. It can further be assumed that $T_r = \partial B_r \cap \partial U$.

Let us recall that a marble is built by gluing two solid tori, one of which contains a closed braid. For each $H_r$, let $T'_r$ be the solid torus of the construction that contains $\beta_r$. By construction, $T'_1, \ldots, T'_l$ are disjoint and unknotted in $\partial T$. Is is trivial to see that, for every $r$, there is a ball $B'_r \subset H_r$ such that $\Phi_r \subset B'_r$ and $T'_r = \partial B'_r \cap \partial T$. Therefore $h(T'_1), \ldots, h(T'_l)$ are disjoint and unknotted in $\partial U$ and satisfy that $h(T'_r) = \partial h(B'_r) \cap \partial U$.

The condition that $T_1, \ldots, T_l$ are unknotted in $\partial U$ implies that $B_1, \ldots, B_l$ can be moved and rearranged freely by means of isotopies of $U$. Let us notice that this would not be true for a three-dimensional ball, but it is for a four-dimensional ball like $U$, because each $B_i$ is homotopically equivalent to a disk intersecting $\partial U$ on its boundary circumference. In particular, and since $h(T'_1), \ldots, h(T'_l)$ are also unknotted, $B_1, \ldots, B_l$ can be taken into $h(B'_1), \ldots, h(B'_l)$.

To see that it is possible to take $\Phi_r$ into $h(\Phi_r)$ by an isotopy of this kind it is enough to see that $\beta_r$ can be taken to $h(\beta_r)$. This can be easily shown by using the fact that every orientation-preserving homeomorphism in $S^3$ is isotopic to the identity. $\square$

*Proof of Lemma 1.11.* Let $(B, \Phi'_1, \ldots, \Phi'_l)$ be a copy of $(T, \Phi_1, \ldots, \Phi_l)$. Let $i_T : B \longrightarrow T$ be the inclusion map (Sending $\Phi'_r$ into $\Phi_r$), and $i'_U : B \longrightarrow U$ an orientation-preserving homeomorphism. Let $\varphi : i_T \longrightarrow i'_U$ be an isotopy in $\mathbb{C}^2$. Let us keep in mind that, for every $r$, $\varphi$ sends $i_T(\Phi'_r) = \Phi_r$ into $i'_U(\Phi'_r)$.

Let $i_U : B \longrightarrow U$ be a homeomorphism such that, for every $r$, $i_U(\Phi'_r) = \Phi_r$. The existence of such a map is implied by Lemma 1.12, because by choosing any $h$, the map $H_1^{-1} \circ h : T \longrightarrow U$ is a homeomorphism sending each $\Phi_r$ into itself. By identifying $B$ with $T$, we obtain $i_U$.

Then, again by Lemma 1.12, and by identifying $B$ with $U$ by means of $i_U$, and defining $h = i'_U \circ i_T^{-1} : T \longrightarrow U$, there exists an isotopy $\bar{\psi} : i_U \longrightarrow i'_U$ that sends each $\Phi_r$ into $h(\Phi_r)$. This is, that sends each $i_U(\Phi'_r)$ into $h(\Phi_r) = i'_U(\Phi'_r)$.

Let $\psi : i'_U \longrightarrow i_U$ be the reverse isotopy of $\bar{\psi}$. Then, by joining $\varphi$ and $\psi$ we obtain an isotopy $\omega : i_T \longrightarrow i_U$. Since $\varphi$ sends each $i_T(\Phi'_r)$ into $i'_U(\Phi'_r)$, and $\psi$ each $i'_U(\Phi'_r)$ into $i_U(\Phi'_r)$, then $\omega$ sends each $i_T(\Phi'_r)$ into $i_U(\Phi'_r)$. This is, for every $r$, $\omega$ sends $\Phi_r$ into $\Phi_r$.

On the other hand, it is not difficult to show that there exists an isotopy $\chi : \mathbb{C}^2 \times I \longrightarrow \mathbb{C}^2$, such that $\chi_0 = \chi_1 = 1_{\mathbb{C}^2}$, and that, for every $t$, $\chi_t(\omega_t(\Phi_r)) = \Phi_r$. The idea is to observe that, for every $r$, $(\omega_t(p_r), t)$ and $(p_r, t)$ form two paths in $\mathbb{C}^2 \times I$, both starting at $(p_r, 0)$ and finishing at $(p_r, 1)$, and therefore forming the continuous image of a circumference that we call $s_r$. Since $\mathbb{C}^2 \times I$ is a five-dimensional space, $s_1, \dots, s_l$ are unknotted, for which there exists a homeomorphism (even an isotopy of the identity) from $\mathbb{C}^2 \times I$ to $\mathbb{C}^2 \times I$, taking each $(\omega_t(p_r), t)$ into $(p_r, t)$. By extending this homeomorphism to each $(\omega_t(\Phi_r), t)$ and $(\Phi_r, t)$ we obtain $\chi$.

Then, $\theta : B \times I \longrightarrow \mathbb{C}^2$ defined by $\theta(x, t) = \chi_t(\omega_t(x))$ is an isotopy from $i_T$ to $i_U$ constant on each $\Phi_r$. $\square$

As a consequence of this, we can think about $T$ as an arbitrary closed ball contained in $D^\varepsilon_{\hat{x}} \times \mathbb{C}$ satisfying I and II. The following theorem is an immediate consequence of Lemma 1.11.

**Theorem 1.13.** Let $a$, $\gamma$, and $\beta$ be as defined in this section. Let $\rho$ be an SCP at $a$ that separates $\beta$, defining the ordered set of sub-braids $\{\beta_{(1)}, \dots, \beta_{(l)}\}$. Let $U$ be a closed four-dimensional ball contained in $D^\varepsilon_{\hat{x}} \times \mathbb{C}$ satisfying that

(I) $U \cap V(f) = V(f) \cap (D^\varepsilon_{\hat{x}} \times \mathbb{C})$ and
(II) $\partial U \cap V(f) = \beta$.

A tower for $\hat{x}$, built upon the ordered set of sub-braids $\{\beta_{(1)}, \dots, \beta_{(l)}\}$, and their factorizations, is a well defined regular CW decomposition for $(U, U \cap V(f))$.

*Proof.* Let us recall the ordered set of sub-braids $\{\beta_{(1)}, \dots, \beta_{(l)}\}$ is defined upon $\rho$. Therefore, to prove the good definition we need to show three things. In the first place, that the topology of the underlying pair of spaces of the tower is independent of $\rho$, in the second place, that it is independent of the chosen factorizations for $\beta_{(1)}, \dots, \beta_{(l)}$ and, in the third place, that this pair is $(U, U \cap V(f))$.

In Lemma 1.11 we showed that the underlying pair of a tower constructed upon an arbitrary $\rho$, and arbitrary factorizations of the sub-braids, is $(U, U \cap V(f))$. This implies the three statements. $\square$

Given a tower as constructed above, for any $1 \leq i \leq k$, the set $\bigcup_{r=1}^{l} C_{(i,r)}$ is a big cylinder with bottom equal to $\bigcup_{r=1}^{l} D_{(i,r)}$ and top equal to $\bigcup_{r=1}^{l} D_{(i+1,r)}$.

**Definition 1.7.** We call the closure of the first of these cylinders, $cl(\bigcup_{r=1}^{l} C_{(1,r)})$, the *stump of $T$*.

This subcomplex will be important for us later, since we will use it to glue $T$ to other complexes.

We will describe now the cells composing $T$ and its boundaries. The set of cells of $T$ is the union of the cells of each of the $H_r$, accounting for the identifications. According to the notation from Section 2, the set of cells of a generic marble $H$ is

$$\mathrm{Bnd}(H) \cup \mathrm{Con}(H) \cup \{AA\},$$

where

$$\begin{array}{rcl} \mathrm{Bnd}(H) & = & \{\mu, \lambda, H^{\sup}, H^{\inf}\} \cup \{\sigma \in C_i\} \text{ and} \\ \mathrm{Con}(H) & = & \{\vee\sigma \mid \sigma \in \mathrm{Bnd}(H)\}\,. \end{array}$$

For each $H_r$, we call the cells of $\mathrm{Bnd}(H_r) \cup \{AA\}$ by its usual names, and add a subindex $1 \leq r \leq l$ to each one to indicate to which $H_r$ it belongs. On the other hand, the subindex $1 \leq i \leq k$ will keep indicating to which $C_i$ the cell belongs or, equivalently, to which factor of $\beta_r$ it is associated. If the cell does not belong to any $C_i$ we will use the symbol "$-$" instead of $i$. For a conical cell in $\mathrm{Con}(H_r)$ we just add the symbol "$\vee$" in front of the name of its base cell. Let us define

$$A := \bigcup_{1 \leq r \leq l} \Big[\mathrm{Bnd}(H_r) \cup \mathrm{Con}(H_r) \cup \{AA_{(-,r)}\}\Big]\,.$$

Then, the set of cells of $T$ is given by the quotient

$$A \diagup \phi,$$

where $\phi$ is an equivalence relation that accounts for the identified cells. Since the gluing of two marbles is always done by subcomplexes of their boundaries, the cells grouped in non-trivial equivalence classes of $\phi$ always belong to $\mathrm{Bnd}(H_r)$ for some $r$. Let us specify which are these cells.

The fact that every $H_r$ is glued to $H_{r+1}$ by the identification of $H^{\sup}_{(-,r)}$ and $H^{\inf}_{(-,r+1)}$ implies that the cells to be identified are those in $cl(H^{\sup}_{(-,r)})$ and $cl(H^{\inf}_{(-,r+1)})$ for $1 \leq r \leq l-1$ or, equivalently, those appearing in $\partial^m H^{\sup}_{(-,r)}$ or $\partial^m H^{\inf}_{(-,r+1)}$ for some $m$. These identifications are exactly the following, taking into account the conditions imposed on the gluing:

I. $H^{\text{sup}}_{(-,r)} = H^{\text{inf}}_{(-,r+1)}$ for $1 \leq r \leq l-1$. For each of these $r$, we will denote the cell resulting from this identification by $H^{\text{inf}}_{(-,r+1)}$. The cell $H^{\text{sup}}_{(-,l)}$, which is the only one of its kind not being identified, will be called $H^{\text{sup}}_{(-,-)}$, since it is no longer dependent on $r$. The following two cases are similar to this one.

II. $\varkappa_{(i,r)} = \kappa_{(i,r+1)}$ for $1 \leq r \leq l-1$. We will denote the cell resulting from this identification by $\kappa_{(i,r+1)}$. The cell $\varkappa_{(i,l)}$ will be called $\varkappa_{(i,-)}$.

III. $m_{2(i,r)} = m_{1(i,r+1)}$ for $1 \leq r \leq l-1$. We will denote the cell resulting from this identification by $m_{1(i,r+1)}$. The cell $m_{2(i,l)}$ will be called $m_{2(i,-)}$.

IV. $\mu_{(-,r)} = \mu_{(-,r+1)}$ for $1 \leq r \leq l$. This implies that all the $\mu_{(-,r)}$ become identified into a single cell. The cell resulting from this identification will be called $\mu_{(-,-)}$ or simply $\mu$. The following five cases are similar to this one.

V. $\lambda_{(-,r)} = \lambda_{(-,r+1)}$ for $1 \leq r \leq l$. The single cell resulting from this identification will be called $\lambda_{(-,-)}$ or simply $\lambda$.

VI. $e_{0(i,r)} = e_{0(i,r+1)}$ for $1 \leq r \leq l$. The single cell resulting from this identification will be called $e_{0(i,-)}$.

VII. $e_{n_r+1(i,r)} = e_{n_{r+1}+1(i,r+1)}$ for $1 \leq r \leq l$. The single cell resulting from this identification will be called $e_{n+1(i,-)}$. The subindex $n+1$ here is rather arbitrary and it was chosen for practical reasons: It is a number, it is independent of $r$, and it is greater than any $n_r$ (which implies that there is no cell previously named $e_{n+1}$).

VIII. $A_{0(i,r)} = A_{0(i,r+1)}$ for $1 \leq r \leq l$. The single cell resulting from this identification will be called $A_{0(i,-)}$.

IX. $A_{n_r+1(i,r)} = A_{n_{r+1}+1(i,r+1)}$ for $1 \leq r \leq l$. The single cell resulting from this identification will be called $A_{n+1(i,-)}$.

Let $B$ be the set of cells of $T$, with the names directly inherited from the marbles (with the added subindex $(r)$) or given in I-IX, according to the case. Let $g : A \longrightarrow B$ be the function that sends each cell to its corresponding cell in $T$.

Let $\rho$ be a cell of some $H_r$ that is being identified to another cell, and let us suppose that $\rho \neq g(\rho)$. Then, this identification can also be thought as the elimination of the cell $\rho$ from the complex and its replacement with $g(\rho)$. This way of thinking will prove useful sometimes, for which we will use this language occasionally. A cell $\rho$ eliminated in this way will be called a *ghost cell*.

We should be careful to notice that, by identifying two cells, their cones never become identified. This means that, for the types of cells listed before, the one-to-one correspondence between conical and not conical cells is lost. In the case of $\mu$, for example, the cells $\mu_{(-,1)}, \ldots, \mu_{(-,l)}$ have been removed from the complex, but their cones $\vee\mu_{(-,1)}, \ldots, \vee\mu_{(-,l)}$

have not. At the same time, a cell $\mu_{(-,-)}$ have been introduced, but for this cell there does not exist a conical cell $\vee\mu_{(-,-)}$.

To end this section we examine the boundaries of the cells of $T$. The boundary of any cell $\rho$ in $T$ is given by the boundary of $\rho$ in $H_r$, where $H_r$ is the marble to which $\rho$ belongs. To find such a boundary we use the explicit formulae we have already given for the boundaries of the cells of a marble, but making the replacements indicated in I to IX, in the case that cells of these types appear. Explicitly, if $\rho$ is a cell of $H_r$ and its boundary there is given by $\partial_{H_r}\rho = \sigma_1 + \cdots + \sigma_p$, then the boundary in $T$ of $g(\rho)$ is given by $\partial_T(g(\rho)) = g(\sigma_1) + \cdots + g(\sigma_p)$. If we extend $g$ linearly to chains, we can write

$$\partial_T(g(\rho)) = g(\partial_{H_r}\rho).$$

In the case of the conical cells this formula can be elaborated a little more and is worth a commentary. Let us recall that for any cell $\rho$ in $H_r$, with $\dim(\rho) > 0$, the boundary in $H_r$ of $\rho$ is given by $\partial_{H_r}(\vee\rho) = (-1)^{\dim(\rho)}(-\rho) + \vee(\partial_{H_r}\rho)$. Then, the boundary in $T$ of $g(\vee\rho)$ is given by

$$\partial_T(g(\vee\rho)) = (-1)^{\dim(\rho)}(-g(\rho)) + g(\vee(\partial_{H_r}\rho)).$$

And, since $g$ always leave conical cells invariant, we obtain the formula

$$\partial_T(\vee\rho) = (-1)^{\dim(\rho)}(-g(\rho)) + \vee(\partial_{H_r}\rho).$$

Let us notice also that $\rho$ is always a cell in $A$, and therefore this formula allows us to calculate the boundary of all the conical cells of $T$, even of those having ghost cells as its bases. The formula also allows us to calculate the boundary of conical cells $\vee\rho$ in $T$ such that $\partial_{H_r}\rho$ contains a ghost cell.

Let us consider the cell $\vee\mu_{(-,1)}$ of $T$ as an example. The boundary in $T$ of this cell is calculated as follows, according to our formula.

$$\begin{aligned}\partial_T(\vee\mu_{(-,1)}) &= -g(\mu_{(-,1)}) + \vee(\partial_{H_1}\mu_{(-,1)}) \\ &= -\mu + \vee e_{0(1,1)} + \cdots + \vee e_{0(k,1)}.\end{aligned}$$

It is worth observing that $\vee\mu_{(-,1)}$ is the cone of a ghost cell, and the boundary of $\mu_{(-,1)}$ (in $H_1$) is composed also of ghost cells. However, the formula provides us the correct boundary of $\vee\mu_{(-,1)}$ in $T$, and it can be checked that all the cells appearing on this boundary exist in $T$.

## 1.6 Joints

At this point we already have CW complexes associated with the local braids. In order to connect these with the complexes associated with the conjugating braids, yet to be constructed, we need to provide a rather peculiar decomposition of a cylinder with a trivial braid, different from the double prism we already defined.

Let us consider a tower $T$, for which we will use all the notation of the previous section. Let us consider the bottom $E := \bigcup_{r=1}^{l} D_{(1,r)}$ of the stump $\bigcup_{r=1}^{l} C_{(1,r)}$ of $T$, which is illustrated in the following figure. Let us embed this disk in $\mathbb{R}^3$ in order that it coincides with the disk with equations $z = 0$, $(x-1/2)^2 + y^2 \leq 1/4$; and in such a way that $A_{0(1,-)} = (0,0,0)$, $A_{n+1(1,-)} = (1,0,0)$, and the points of $m_{1(1,1)}$ have non-positive $y$ coordinate.

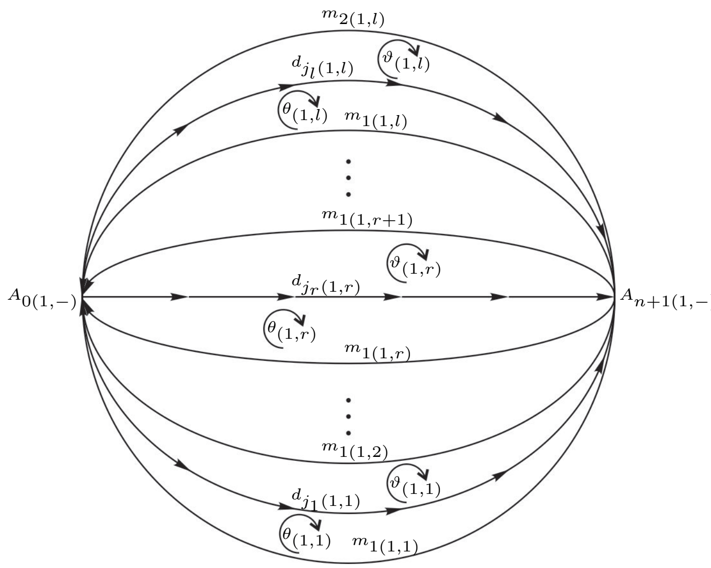


**Figure 1.10**

Let us also consider the bottom $F$ a double prism of $n$ strands, and let us embed this disk in $\mathbb{R}^3$ in order that it coincides with the disk with equations $z = 1$, $(x+1/2)^2 + y^2 \leq 1/4$; and in such a way that $A_0 = (0,0,1)$, $A_{n+1} = (1,0,1)$, and the points of $m_1$ have non-positive $y$ coordinate. Here the vertices $A_j$, that lie in $F$, should not be confused with the vertices $A_{j(i,r)}$ and $A_{j(i,-)}$, that lie in $E$ and come from $T$ . Let $C'$ be the cylinder encompassed between $E$ and $F$. We now define a set of edges that join each vertex on $E$ with a vertex on $F$. For $1 \leq r \leq l$ let $\Sigma_r$ be defined by

$$n_0 := 0, \;\; \Sigma_r := \sum_{s=0}^{r} n_s.$$

For $1 \leq r \leq l$ and $1 \leq j_r \leq n_r$ let us define the following objects, which can be seen in Figures 1.11 and 1.12.

- Let $z_{r,j_r}$ be the segment running from $A_{j_r(1,r)}$ in $E$ to $A_{\Sigma_{r-1}+j_r}$ in $F$ through the cylinder $C'$.

- Let $w0$ be the segment running from $A_{0(1,-)}$ in $E$ to $A_{\Sigma_0} = A_0$ in $F$.
- Let $w1$ be the segment running from $A_{n+1(1,-)}$ in $E$ to $A_{\Sigma_l+1} = A_{n+1}$ in $F$.

Let us order the set $\{A_{\Sigma_r+j_r}\}$ by the natural order of its subindices, and the set $\{A_{j_r(1,r)}\}$ by the lexicographical order defined by $A_{j_r(1,r)} \leq A_{j_{r'}(1,r')}$ if and only if $r \leq r'$, or $r = r'$ and $j_r \leq j_{r'}$. Then, we can naturally join the points of each of these sets according to this ordering in such a way that the segments $\{z_{r,j_r}\}$ are the $n$ strands of the trivial braid $e$ of $\mathcal{B}_n$.

We want to fill $C'$ with cells in order that the segments $z_{r,j_r}$, $w0$ and $w1$ are all subcomplexes of the resulting complex, and without modifying the CW decompositions that we already have within $E$ and $F$. The idea of the construction is the following. Let us notice that the edges $d_0, \dots, d_n$ form a diameter of $F$ that runs from $A_0$ to $A_{n+1}$. Similarly, for each $1 \leq r \leq l$, the edges $d_{0(1,r)}, \dots, d_{n_r(1,r)}$ form a path in $E$ joining $A_{0(1,-)}$ with $A_{n+1(1,-)}$. We will introduce $l$ quadrilaterals $\chi_r$, that we depict as rectangles (Fig. 1.11), satisfying the following:

1. Each $\chi_r$ have $d_{0(1,r)} \cup \cdots \cup d_{n_r(1,r)}$ as its base, but all of them share $w0$ as right side, $w1$ as left side, and $d_0 \cup \cdots \cup d_n$ as upper side.
2. For $1 \leq r \leq l$, the strands $z_{r,j_r}$ run through the interior of $\chi_r$.

Since the rectangles $\chi_1, \dots, \chi_l$ split the interior of $C'$ into $l+1$ three-dimensional balls, they induce a CW decomposition of $C'$ of which the already given decompositions of $E$ and $F$ are subcomplexes.

We will construct now this decomposition, by defining each of the cells in $C' \setminus (E \cup F)$. The following figure illustrates a rectangle $\chi_r$ with the cells that compose it, and that we are about to define.

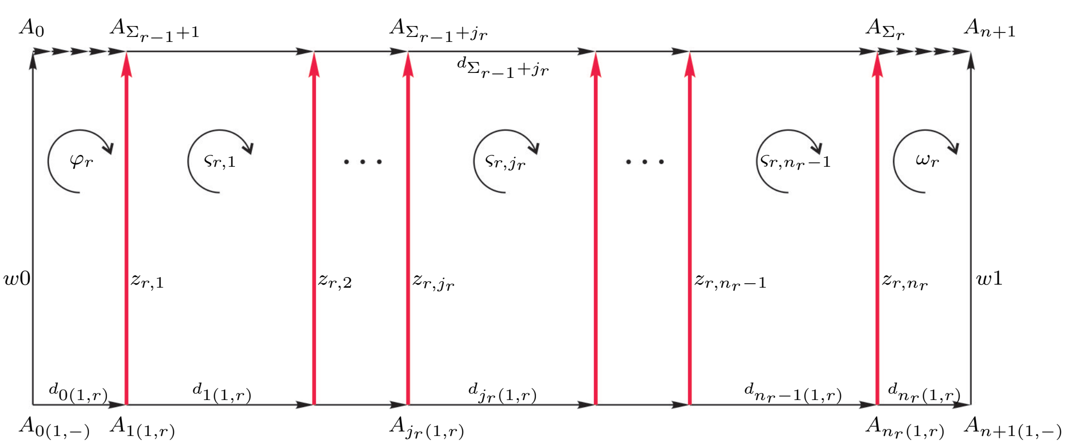


**Figure 1.11**

- We take $w0$, $w1$ and all the segments in $\{z_{r,j_r}\}$ as the one-dimensional cells.
- Let us notice that $w0$ and $w1$ split $\partial C' \backslash (E \cup F)$ into two cells. We call $\xi$ the cell containing negative $y$ coordinates and $\psi$ the one containing positive ones.
- For $1 \leq r \leq l$ and $0 \leq j_r \leq n_r - 1$ let us define $\zeta_{r,j_r}$ as a quadrilateral with boundary $z_{r,j_r} + d_{\Sigma_{r-1}+j_r} - z_{r,j_r+1} - d_{j_r(1,r)}$. This quadrilateral serve to connect $z_{r,j_r}$ with $z_{r,j_r+1}$ without interfering with the decompositions of $E$ and $F$.
- For $1 \leq r \leq l$, let $\varphi_r$ be the quadrilateral bounded by $d_0 + \cdots + d_{\Sigma_{r-1}} - z_{r,1} - d_{0(1,r)} + w0$, and $\omega_r$ the one bounded by $d_{\Sigma_{n_r}} + \cdots + d_n - w1 - d_{n_r(1,r)} + z_{r,n_r}$. These quadrilaterals serve, the first to connect $w0$ with $z_{r,1}$, and the second to connect $w1$ with $z_{r,n_r}$.
- Then, for each $1 \leq r \leq l$, the quadrilaterals $\varphi_r, \zeta_{r,1}, \ldots, \zeta_{r,n_r-1}, \omega_r$ join consecutively, in this order, to form a bigger quadrilateral $\chi_r$ that contains the strands $z_{r,1}, \ldots ,$ $z_{r,n_r}$. The $l+1$ open balls in which $C'$ is splitted by $\chi_1, \ldots, \chi_l$ will be called $\Xi$, $\Lambda_1$, $\ldots$ , $\Lambda_{l-1}$, and $\Psi$, from lesser to greater $y$ coordinates.

By this process we have constructed a CW decomposition for $(C', e)$ respecting the original decompositions of $E$ and $F$. Let us notice that the resulting CW complex is built only from $E$, or more precisely, from the order and number of strands of the sub-braids $\beta_{(r)}$, i.e. the ordered list $(n_1, \ldots, n_l)$.

**Definition 1.8.** We call $C'$ endowed with this CW complex structure a *joint* for $T$ or, equivalently, for $E$ or for $(n_1, \ldots, n_l)$.

This complex is shown in Figures 1.11 and 1.12. with names and orientations for each cell. The following theorem is clear by construction.

**Theorem 1.14.** Let $T$ be a tower and $(n_1, \ldots, n_l)$ the list of the number of strands of the sub-braids $\beta_{(r)}$ by order. Let $E$ and $F$ be as defined in this section. A joint for $T$ (or for $(n_1, \ldots, n_l)$) is a well defined regular CW decomposition for $(C', e)$ that has $E$ and $F$ as subcomplexes.

We will give now the boundaries of the cells composing $(C', e)$. Let us notice that the cells of $(C', e)$ are divided into three sets: the cells of $E$, the cells of $F$, and the cells of $C' \backslash (E \cup F)$. The boundary of any cell of $E$ is given by its boundary in $T$, which is already known. Similarly, the boundary of any cell of $F$ is given by its boundary in a double prism or quasi-prism with bottom $F$. The boundaries of the remaining cells, those of $C' \backslash (E \cup F)$, are given below.

Let us notice that a joint of this kind can also be constructed taking $E = \bigcup_{r=1}^{l} D_{(s,r)}$ (for any $s$) as the bottom of $C'$. In that case, we just replace in the following formulae the subindex $i = 1$ for $i = s$ in all the cells belonging to $E$.

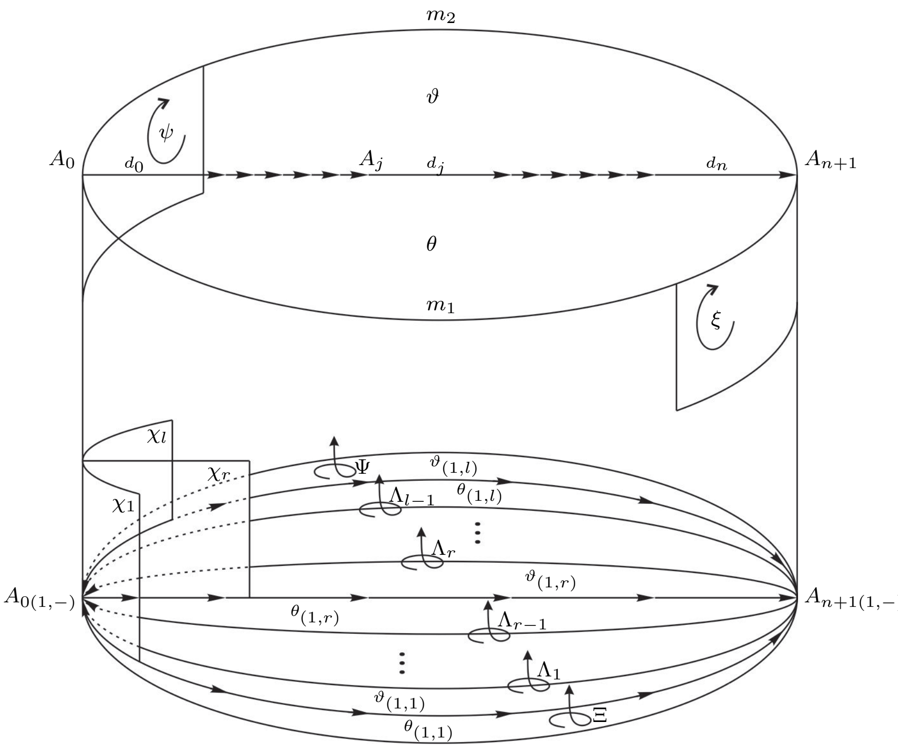


**Figure 1.12**

Dim. 1

$$\partial(w0) = A_0 - A_{0(1,-)}$$
$$\partial(w1) = A_{n+1} - A_{n+1(1,-)}$$

for $1 \leq r \leq l$, $1 \leq j_r \leq n_r$:

$$\partial(z_{r,j_r}) = A_{\Sigma_{r-1}+j_r} - A_{j_r(1,r)}$$

Dim. 2

$$\partial(\xi) = w0 - m_1 - w1 + m_{1(1,1)}$$
$$\partial(\psi) = w0 - m_2 - w1 + m_{2(1,l)}$$

$$\partial(\varphi_r) = w0 - z_{r,1} - d_{0(1,r)} + \sum_{j=0}^{\Sigma_{r-}} d_j$$

$$\partial(\omega_r) = -w1 + z_{r,n_r} - d_{n_r(1,r)} + \sum_{j=\Sigma_r}^{n} d_j$$

For $1 \leq r \leq l$, $1 \leq j_r \leq n_r - 1$:

$$\partial(\zeta_{r,j_r}) = z_{r,j_r} - z_{r,j_r+1} + d_{\Sigma_{r-1}+j_r} - d_{j_r(1,r)}$$

Dim. 3

$$\partial(\Xi) = \xi - \theta_{(1,1)} + \theta - \varphi_1 - \sum_{j_1=1}^{n_1-1} \zeta_{1,j_1} - \omega_1$$
$$\partial(\Psi) = -\psi - \vartheta_{(1,l)} + \vartheta + \varphi_l + \sum_{j_l=1}^{n_l-1} \zeta_{l,j_l} + \omega_l$$

For $1 \leq r \leq l-1$:
$$\partial(\Lambda_r) = -\vartheta_{(1,r)} - \theta_{(1,r+1)} + \varphi_r + \sum_{j_r=1}^{n_r-1} \zeta_{r,j_r} + \omega_r - \varphi_{r+1} - \sum_{j_{r+1}=1}^{n_{r+1}-1} \zeta_{r+1,j_{r+1}} - \omega_{r+1}$$

## 1.7 Decomposition for a Conjugating Braid

Let us consider $f$ and $\Omega$ once more. Our aim now is to construct a CW complex, analogous to the complex already constructed for a local braid, but associated with a conjugating braid or, more generally, with an arbitrary open path.

As in the first section, let $\lambda$ be a simple curve in $\mathbb{C}\backslash\Delta$ with initial point $b$ and final point $a$, and let $\alpha$ be the braid associated to $\lambda$. In the case that interests us we will take $b$ and $a$ as certain points close to $x_0$ and some $x_s$ respectively, though in this section we consider $\lambda$ as a general path in $\mathbb{C} \setminus \Delta$. Let us swell $\lambda$ a little to form a narrow strip $\bar{\lambda}$. We may define $\bar{\lambda}$ as the image of an injective isotopy $Is : \lambda \times I \longrightarrow \mathbb{C}\backslash\Delta$ with $Is(\lambda \times \{0\}) = \lambda$, where $I = [0,1]$. For every $t$, we denote $Is(\lambda \times \{t\})$ by $\lambda_t$. Since this isotopy is an embedding, $\bar{\lambda}$ trivially inherits the product structure of $\lambda \times I$. For simplicity, we write $\bar{\lambda} = \lambda \times I$ and $\lambda_t = \lambda \times \{t\}$.

Let $V_{\bar{\lambda}}$ be the set of points of $V(f)$ with first coordinate belonging to $\bar{\lambda}$. Then, since the isotopy $Is$ is small enough, $V_{\bar{\lambda}}$ has the product structure $V_{\bar{\lambda}} = \alpha \times I$, where every fiber $\alpha \times \{t\}$ is defined as the set of points of $V(f)$ with first coordinate belonging to $\lambda_t$. We denote $\alpha \times \{t\}$ by $\alpha_t$. Notice that $\alpha_0 = \alpha$.

Let $X \in \mathbb{C}$ be defined by $X := \left\{ y \in \mathbb{C} \mid (x,y) \in V_{\bar{\lambda}} \text{ for some } x \in \bar{\lambda} \right\}$.

**Claim 1.15.** *The set $X$ is bounded.*

*Proof.* Let us suppose that $X$ is not bounded. Then, there exists a sequence $\{(x_m, y_m)\}$ in $V_{\bar{\lambda}}$ such that $\|y_m\| \to \infty$. Since $\bar{\lambda}$ is compact, $\{x_m\}$ has a subsequence $\{x_{m_i}\}$ convergent to some point $x \in \bar{\lambda}$. Let us consider now the subsequence $\{(x_{m_i}, y_{m_i})\}$ of $\{(x_m, y_m)\}$, which still satisfies that $\|y_{m_i}\| \to \infty$. This divergence along with the continuity of $f$ implies that $f$ has less than $n$ roots at $x$. This is not possible since $x \in \bar{\lambda} \subset \mathbb{C}\backslash\Delta$. $\square$

Let $D \subset \mathbb{C}$ be a disk containing $X$. We will now take the spaces $\bar{\lambda} \times D$ and $V_{\bar{\lambda}}$ into consideration. Since $\bar{\lambda} = \lambda \times I$, then $\bar{\lambda} \times D$ has the product structure $\bar{\lambda} \times D = \lambda \times I \times D$. It follows, by the definitions of $V_{\bar{\lambda}}$, $\alpha_t$ and $D$, that

$$\begin{aligned} V_{\bar{\lambda}} &= (\bar{\lambda} \times \mathbb{C}) \cap V(f) = (\bar{\lambda} \times D) \cap V(f) \text{ and} \\ \alpha_t &= (\lambda_t \times \mathbb{C}) \cap V(f) = (\lambda_t \times D) \cap V(f). \end{aligned}$$

The situation is illustrated in the following figure.

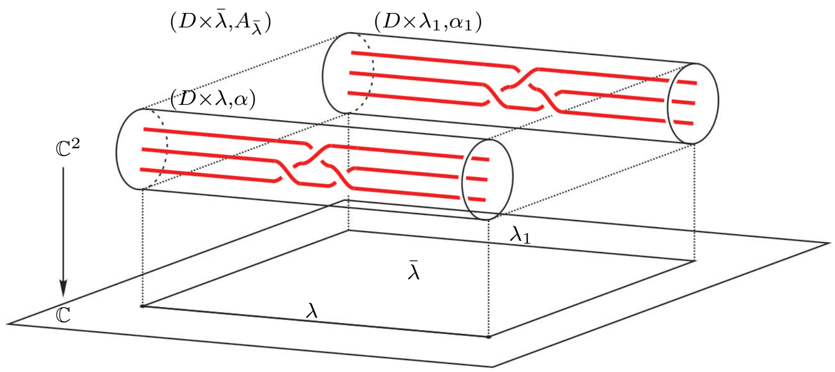


**Figure 1.13**

Let us observe now that, for every $t$, $\lambda \times \{t\} \times D$ is a cylinder in which $\alpha_t$ is embedded. In particular, $\lambda \times \{0\} \times D$ is a cylinder that contains the braid $\alpha$, which is the situation of the first section. Now, we construct a CW decomposition for $(\lambda \times \{0\} \times D, \alpha)$ and afterwards for $(\bar{\lambda} \times D, V_{\bar{\lambda}})$.

Let us choose two points $x_0$ and $x_s$ in $\Delta$, and consider towers $T_{x_0}$ and $T_{x_s}$ for $x_0$ and $x_s$. Let us choose also a factorization $\alpha = \tau_1 \cdot \ldots \cdot \tau_{kc}$ of $\alpha$, and add the redundant factor $e$ at the beginning and at the end, obtaining $\alpha = e\tau_1 \cdot \ldots \cdot \tau_{kc}e$. Then, we divide $\lambda \times \{0\} \times D$ into sub-cylinders $C_0, \ldots, C_{kc+1}$ as in the first section, each sub-cylinder $C_i$ having top $D_i$ and bottom $D_{i-1}$, and containing the corresponding factor of $\alpha$. Now, we endow the cylinder $C_1 \cup \cdots \cup C_{kc}$ with a loom structure applying Theorem 1.2. Finally, we endow the cylinders $C_0$ and $C_{kc+1}$ with joint structures associated with $T_{x_0}$ and $T_{x_s}$ respectively.

In this way we have a CW structure for $(\lambda \times \{0\} \times D, \alpha)$. Now we extend such structure to $(\bar{\lambda} \times D, A_{\bar{\lambda}})$ by means of the product structure $\lambda \times I \times D$ of $\bar{\lambda} \times D$. Let us notice that the resulting CW complex is built only from a factorization $\tau_1 \cdot \ldots \cdot \tau_{kc}$ of $\alpha$ and from towers $T_{x_0}$ and $T_{x_s}$ (Or more precisely, from the ordered list $(n_1, \ldots, n_l)$ for $x_0$ and its counterpart for $x_1$).

**Definition 1.9.** We call $\lambda \times \{0\} \times D$ endowed with this CW complex structure a *bridge* for $\alpha = \tau_1 \cdot \ldots \cdot \tau_{kc}$, $T_{x_0}$ and $T_{x_s}$.

**Theorem 1.16.** Let $\alpha$ be as defined in this section, let $\tau_1 \cdot \ldots \cdot \tau_{kc}$ be a factorization of $\alpha$, and $T_{x_0}$ and $T_{x_s}$ towers for $x_0$ and $x_s$. A bridge for $\alpha = \tau_1 \cdot \ldots \cdot \tau_{kc}$, $x_0$ and $x_s$ is a well defined regular CW decomposition for $(\bar{\lambda} \times D, V_{\bar{\lambda}})$, were $\bar{\lambda}$, $D$ and $A_{\bar{\lambda}}$ are constructed as before.

Proof. It is clear by construction and by Theorems 1.2 and 1.14. □

**Definition 1.10.** Given a bridge as constructed above, the cylinders $a \times I \times D$ and $b \times I \times D$, will be called respectively the *initial end* and *final end* of $\bar{\lambda} \times D$.

Similarly, the cylinders $\lambda \times \{0\} \times D$ and $\lambda \times \{1\} \times D$ will be called respectively the *bottom* and *top* of $\bar{\lambda} \times D$, and denoted by $\mathrm{Bot}(\bar{\lambda} \times D)$ and $\mathrm{Top}(\bar{\lambda} \times D)$.

We will give now the boundaries of the cells composing $\bar{\lambda} \times D$. Let us notice that the cells of $\bar{\lambda} \times D$ are divided into three sets: the cells of $\mathrm{Bot}(\bar{\lambda} \times D)$, the cells of $\mathrm{Top}(\bar{\lambda} \times D)$, which are an exact copy of the former, and the product cells $\mathrm{Prod}(\bar{\lambda} \times D)$ produced by taking each cell of $\mathrm{Bot}(\bar{\lambda} \times D)$ and multiplying it by $I$.

We begin with the cells of $\mathrm{Bot}(\bar{\lambda} \times D)$, which are themselves divided into the cells of $C_1 \cup \cdots \cup C_{kc}$ and the cells of $C_0$ and $C_{kc+1}$. Since $C_1 \cup \cdots \cup C_{kc}$ is a loom, the boundaries of its cells are as described in the first section. The boundaries of the cells of $C_0$ and $C_{kc+1}$ are as described in the previous section. The cells in $\mathrm{Bot}(\bar{\lambda} \times D)$ are given the names already established in the first section and the previous section, but adding the subindex $(i, 0)$ to indicate to which $C_i$ they belong.

On the other hand, $\mathrm{Top}(\bar{\lambda} \times D)$ is a copy of $\mathrm{Bot}(\bar{\lambda} \times D)$ and the boundaries of their cells are the same. The cells in $\mathrm{Top}(\bar{\lambda} \times D)$ are given the names already established in the first and last sections, but adding the subindex $(i, 1)$ to indicate to which $C_i$ they belong.

Finally, we consider the product cells. For any cell $\rho \in \mathrm{Bot}(\bar{\lambda} \times D)$, let $I\rho$ denote the product cell $\rho \times I$. Moreover, for any chain $c = \sigma_1 + \cdots + \sigma_l$ of cells of a given dimension in $\mathrm{Bot}(\bar{\lambda} \times D)$, let $Ic$ denote the chain $I\sigma_1 + \cdots + I\sigma_l$. We give each cell $I\rho$ the orientation resulting by adding to the orientation of $\rho$ the direction running from $\mathrm{Bot}(\bar{\lambda} \times D)$ to $\mathrm{Top}(\bar{\lambda} \times D)$. Then, the boundaries of the product cells are given by

$$\partial(I\rho) = \rho_1 - \rho_0$$

if $\dim(\rho) = 0$, and by

$$\partial(I\rho) = (-1)^{\dim(\rho)}(\rho_1 - \rho_0) + I(\partial(\rho))$$

if $\dim(\rho) < 0$.

## 1.8 Global Decomposition

Let us return to our goal of obtaining a complete topological description of the embedding of $\Omega$ in $\mathbb{C}^2$. Let us recall that $\Omega$ is defined by a function $f : \mathbb{C}^2 \to \mathbb{C}$ of the form

$$f(x, y) = y^n + a_1(x)y^{n-1} + \cdots + a_{n-1}(x)y + a_n(x),$$

where, for every $i$, $a_i(x) \in \mathbb{C}[x]$ with $\deg(a_i(x)) \leq i$ or $a_i(x) \equiv 0$. Let us recall also that we have defined

$$\Delta = \{x \in \mathbb{C} \mid f(x,y) \text{ has multiple roots}\} = \{x_1, \ldots, x_m\}.$$

Let $D \times E$ be a polydisc such that

$$\begin{array}{rcl} \Delta & \subset & D, \\ \partial(D \times E) \cap \Omega & \subset & \partial D \times E. \end{array}$$

We call such a polydisc a *great polydisc* for $f$. The fact that $\Delta \subset D$ implies moreover that the intersection of $\partial D \times E$ with $\Omega$ is transverse. The two conditions imply also that all the points of the form $(x_s, y) \in V(f)$, with $x_s \in \Delta$, belong to the polydisc.

We will provide now a CW decomposition of the pair $(\mathcal{D}, C \cap \mathcal{D})$, where $\mathcal{D}$ is a great polydisc for $f$. By doing so we obtain a complete topological description of the embedding of $C$ into $\mathbb{C}^2$.

We construct this decomposition by taking towers for every point $x_s \in \Delta$ and a base point, and then connecting the towers through bridges. The steps of the construction are the following:

1. First we define disks $D^\varepsilon_{x_0}, \ldots, D^\varepsilon_{x_m}$ and local braids $\beta_0, \ldots, \beta_l$ around the $m$ points $x_1, \ldots, x_l$ of $\Delta$ and a base point $x_0$.
2. We construct towers $T_{x_0}, \ldots, T_{x_m}$ for $x_0, \ldots, x_l$.
3. We deform these towers slightly for technical reasons.
4. We define paths $\lambda_1, \ldots, \lambda_m$ joining each disk $D^\varepsilon_{x_s}$ to the disk $D^\varepsilon_{x_0}$, and conjugating braids $\alpha_1, \ldots, \alpha_m$ associated to them. We also swell the paths into strips $\bar{\lambda}_1, \ldots, \bar{\lambda}_m$ that fit correctly with $D^\varepsilon_{x_0}, \ldots, D^\varepsilon_{x_m}$.
5. We define bridges $B_1, \ldots, B_m$ for $\lambda_1, \ldots, \lambda_m$.
6. We deform these bridges in order that every $B_s$ fit with $T_{x_s}$ and $T_{x_0}$ correctly.
7. Finally, we deform the resulting ball into a great polydisc.

Let $x_0 \in \mathbb{C} \backslash \Delta$ and $\varepsilon > 0$ such that the disks $D^\varepsilon_{x_0}, \ldots, D^\varepsilon_{x_m}$ are pairwise disjoint. For every $0 \leq s \leq m$, let $\gamma_s$ be the curve given by $\gamma_s(t) = x_s + \varepsilon e^{2i\pi t}$, let $\beta_s$ be the local braid of $x_s$ along $\gamma_s$, and let $\rho^s$ be an SCP at $x_s + \varepsilon$ that separates $\beta_s$. Then, each $\rho^s$ defines an ordered set $\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}$ of sub-braids of $\beta_s$, where $l_s$ is the cardinal of $A_{x_s}$.

Let $T_{x_0}, \ldots, T_{x_m}$ be towers for $x_0, \ldots, x_m$, with each $T_{x_s}$ constructed upon the ordered set $\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}$. Then, each $T_{x_s}$ satisfies that $T_{x_s} \subset D^\varepsilon_{x_s} \times \mathbb{C}$ and the hypotheses of Theorem 1.13.

Lets us recall that each $T_{x_s}$ is constructed from $l_s$ piled marbles $H^s_1, \ldots, H^s_{l_s}$ corresponding to $\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}$ respectively. Each $H^s_r$ in turn possesses a collection of cylinders

$\{C^s_{i(r)}\}$ and a collection of disks $\{D^s_{i(r)}\}$ satisfying that, for every $i$, $D^s_{i-1(r)}$ and $D^s_{i(r)}$ are the bottom and top of $C^s_{i(r)}$. Let us recall also that the union $\bigcup_{r=1}^{l_s} C^s_{1(r)}$ has been called the stump of $T_{x_s}$. We denote this set by $F_s$.

In order to facilitate our construction we perform a slight deformation on each $T_{x_s}$ in the following way. For each $1 \leq s \leq m$ (but not for $s = 0$), we deform $T_{x_s}$ isotopically in order that the bottom and top of $F_s$ (i.e. the disks $\bigcup_{r=1}^{l_s} D^s_{0(r)}$ and $\bigcup_{r=1}^{l_s} D^s_{1(r)}$) are contained in $L_{x_s+\varepsilon}$ and $L_{x_s-i\varepsilon}$ respectively. In fact, to demand only that $(\bigcup_{r=1}^{l_s} D^s_{0(r)}) \cap \beta_s \subset L_{x_s+\varepsilon}$ and $(\bigcup_{r=1}^{l_s} D^s_{1(r)}) \cap \beta_s \subset L_{x_s-i\varepsilon}$ would be enough to our purposes, but we can do it either way.

We also deform $T_{x_0}$ in a similar way. Since $x_0 \notin \Delta$, $A_{x_0}$ is a set of $n$ points and $l_0 = n$. Therefore, $\beta_0$ is a trivial braid of $n$ strands and every $\beta^0_{(r)}$ is a trivial braid of one strand. We impose over each marble $H^0_r$ of $T_{x_0}$ the condition to be associated to the factorization

$$b^0_{(r)} = \underbrace{eee\cdots e}_{m \text{ times}}$$

of $b^0_{(r)}$, where $e$ is the identity of $\mathcal{B}_1$. Then, we deform $T_{x_0}$ in order that, for every $0 \leq p \leq m$, the disk $\bigcup_{r=1}^{n} D^0_{p(r)}$ is contained in $L_{x_0+\varepsilon e^{2i\pi(p/m)}}$. This means that the $x$ coordinate of every point of any $\bigcup_{r=1}^{n} D^0_{p(r)}$ is exactly the $p$-th vertex of a $m$-sided polygon inscribed in $D^\varepsilon_{x_0}$. We call the cylinder $\bigcup_{r=1}^{n} C^0_{p(r)}$ the *p-th stump* of $T_{x_0}$ and denote it by $F_0^{p\text{'th}}$. Let us notice that $F_0^{p\text{'th}}$ lies between $\bigcup_{r=1}^{n} D^0_{p-1(r)} \subset L_{x_0+\varepsilon e^{2i\pi(p-1/m)}}$ and $\bigcup_{r=1}^{n} D^0_{p(r)} \subset L_{x_0+\varepsilon e^{2i\pi(p/m)}}$.

Therefore, $T_{x_0}$ is symmetric by rotations by $2\pi/m$. To ensure later that the towers and bridges fit correctly we need also to define a convenient system of SCP around $x_0$. For every $1 \leq s \leq m$, let $\omega^s$ be an SCP for $x_0 + \varepsilon e^{2i\pi(s/m)}$, i.e., for each vertex of the $m$-sided polygon inscribed in $D^\varepsilon_{x_0}$. By taking $\gamma_0$ narrow enough, we can choose each $\omega^s$ in such a way that it is compatible with $\rho^0$, this is, a translation of $\rho^0$ along $\gamma_0$. In particular, we define $\omega^0 = \rho^0$.

Now we can proceed to the construction of the bridges. For every $1 \leq s \leq m$, let $\lambda_s$ be a simple path

$$\lambda_s : [0,1] \longrightarrow \mathbb{C}\setminus \bigcup_{p=0}^{m} \mathring{D}^\varepsilon_{x_p}$$

satisfying that

$$\begin{aligned} \lambda_s(0) &= x_o + \varepsilon e^{2i\pi(s-1/m)}, \\ \lambda_s(1) &= x_s + \varepsilon, \\ \lambda_s((0,1)) &\subset \mathbb{C}\setminus \bigcup_{p=0}^{m} D^\varepsilon_{x_p}. \end{aligned}$$

We also demand that $\lambda_1, \ldots, \lambda_m$ be disjoint. On the other hand, for every $1 \leq s \leq m$, let

$$\psi_s : L_{x_0+\varepsilon e^{2i\pi(s-1/m)}} \longrightarrow L_{x_s+\varepsilon}$$

be the homeomorphism sending $\omega^s$ into $\rho^s$. Then, as in the previous section, and for every $1 \leq s \leq m$, $\lambda_s$ and $\psi_s$ define a braid $\alpha_s$.

Now we swell each $\lambda_s$ a little to form a narrow strip $\bar{\lambda}_s$ in $\mathbb{C}\setminus\bigcup_{p=0}^{m}\mathring{D}_{x_p}^{\varepsilon}$, with a product structure $\bar{\lambda}_s = \lambda_s \times I$ as in the previous section. We do this in such a way that the following two conditions hold.

1. That $\lambda_{s,1} = \lambda_s \times \{1\}$ is the curve starting at $x_0 + \varepsilon e^{2i\pi(s/m)}$ and ending at $x_s - i\varepsilon$.

2. That the arch $\bar{\lambda}_s \cap \partial D_{x_0}^{\varepsilon}$ (respectively $\bar{\lambda}_s \cap \partial D_{x_s}^{\varepsilon}$) is equal to the fiber

$$(x_0 + \varepsilon e^{2i\pi(s-1/m)}) \times I \qquad (\text{res. } \bar{\lambda}_s \cap \partial D_{x_s}^{\varepsilon} = (x_s + \varepsilon) \times I)$$

   in the product structure $\bar{\lambda}_s = \lambda_s \times I$.

The situation is illustrated in the following figure.

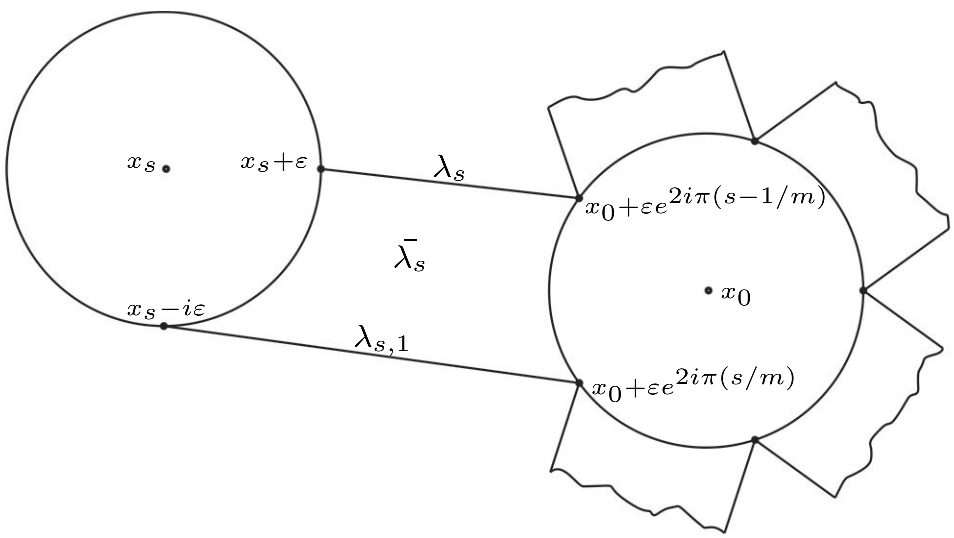


**Figure 1.14**

Now we have defined everything that is needed for the construction of our bridges. For every $1 \leq s \leq m$, let $B_s := \bar{\lambda}_s \times D_s$ be a bridge for $\alpha_s$, $T_{x_0}$ and $T_{x_s}$. Let $In._s$ and $Fin._s$ be the initial and final ends of $B_s$, given by $(x_0 + \varepsilon e^{2i\pi(s-1/m)}) \times I \times D_s$ and $(x_s + \varepsilon) \times I \times D_s$ respectively.

To ensure the correct fitting of $B_s$ with $T_{x_0}$ and $T_{x_s}$ we deform it order that

$$\begin{aligned} B_s \cap T_{x_0} &= In._s = F_0^{s\text{'th}} \text{ and} \\ B_s \cap T_{x_s} &= Fin._s = F_s. \end{aligned}$$

We do this in such a way that $In._s$ and $F_0^{s\text{'th}}$ (res. $Fin._s$ and $F_s$) not only coincide as sets but further as CW complexes. The fact that this can be done is trivial, because the joint and bridge complexes were constructed specifically to ensure that $Fin._s$ and $F_s$ (res.$In._s$ and $F_0^{s\text{'th}}$) are isomorphic subcomplexes. In this way, the initial end of $B_s$ becomes identified with the $s$-th stump of $T_{x_0}$, and the final end with the stump of $T_{x_s}$, effectively connecting $T_{x_0}$ with $T_{x_s}$.

Now let us consider the union

$$G := \left(\bigcup_{s=0}^{m} T_{x_s}\right) \cup \left(\bigcup_{s=1}^{m} B_s\right)$$

with the CW complex structure induced by all of their components. Let us notice that this complex depends only on the braids $\alpha_1, \ldots, \alpha_m$, the ordered families of braids $\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}$, and factorizations of all of these.

**Definition 1.11.** We call $G$ endowed with this CW complex structure a *global decomposition* for $\Omega$.

**Theorem 1.17.** Let $\alpha_1, \ldots, \alpha_m$ and $\{\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}\}_{s=1}^m$ be as defined in this section. A global decomposition, as defined in this section, built upon the braids $\alpha_1, \ldots, \alpha_m$ and $\{\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}\}_{s=1}^m$, and factorizations of all of these, is a well defined regular CW decomposition for $(G, G \cap \Omega)$.

*Proof.* We have already proved in Theorems 1.13 and 1.16 the good definition of the towers and the bridges. It only remains to show that the complex produced by its gluing is a well defined decomposition for $(G, G \cap \Omega)$.

Let us observe that, for any given $1 \leq s \leq m$, it holds that $Fin._s \cap \Omega = F_s \cap \Omega$. Since $Fin._s$ and $F_s$ are three-dimensional balls, and $F_s \cap \Omega$ is a set of $n$ segments, they can be trivially isotoped into one another. That their CW complex structures also coincide is due to the fact that the SCP $\rho^s$ at $x_s + \varepsilon$ used to define $\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}$ is the same one used to define $\alpha_s$. Therefore, their CW structures coincide by definition. We reason in a similar way for $In._s$ and $F_0^{s\text{'th}}$. □

Let us see now that the pair $(G, C \cap G)$ is topologically equivalent to the $(\mathcal{D}, C \cap \mathcal{D})$, for which we have in fact constructed a CW decomposition for the latter.

**Theorem 1.18.** Let $\alpha_1, \ldots, \alpha_m$ and $\{\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}\}_{s=1}^m$ be as defined in this section. Let $\mathcal{D}$ be a great polydisc for $f$. A global decomposition, as defined in this section, built upon $\alpha_1, \ldots, \alpha_m$ and $\{\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}\}_{s=1}^m$, and factorizations of all of these, is a well defined regular CW decomposition for $(\mathcal{D}, C \cap \mathcal{D})$.

*Proof.* Let us define a set $A$ by

$$A = \left(\bigcup_{s=0}^{m} D^{\varepsilon}_{x_s}\right) \cup \left(\bigcup_{s=1}^{m} \bar{\lambda}_s\right),$$

which is a topological disk. Let $D \times E$ be a great polydisc for $f$ such that $A \subset D$. Then, by Theorem 1.13, we may assume that each tower $T_{x_s}$ is a polydisc of the form $D^{\varepsilon}_{x_s} \times E$. Also, by the definition of a bridge, we may assume that each bridge $B_s$ is of the form $\bar{\lambda}_s \times E$.

From here we may assume that $G = A \times E$. Therefore, an isotopy from $A$ to $D$ induces an isotopy from $G$ to $D \times E$. It only remains to see that this isopoty can be chosen to be an isotopy from the pair $(G, G \cap \Omega)$ to the pair $((D \times E), (D \times E) \cap \Omega)$.

Let us define

$$\begin{array}{rcl} X & = & \mathbb{C}^2 \setminus \bigcup_{1 \leq s \leq m} L_{x=x_s}, \\ X' & := & (D \times E) \setminus G. \end{array}$$

Then, the projection

$$\begin{aligned} \pi_x : (X, X \cap \Omega) &\longrightarrow C \\ (x, y) &\longmapsto x \end{aligned}$$

is a fiber bundle of a pair. Therefore, the restriction of $\pi_x$ to the pair $(X', X' \cap \Omega)$ is also a fiber bundle of a pair. This implies that $X'$ possesses the product structure

$$X' = ((\partial D \times E), (\partial D \times E) \cap \Omega) \times I,$$

and also that an isotopy from the pair $(G, G \cap \Omega)$ to the pair $((D \times E), (D \times E) \cap \Omega)$ exists. $\square$

By a similar argument, we can show that $(\mathbb{C}^2 \setminus \mathcal{D}, \Omega \cap (\mathbb{C}^2 \setminus \mathcal{D}))$ is homeomorphic to $(\partial \mathcal{D}, \Omega \cap \partial \mathcal{D}) \times [0, 1)$. Then the CW decomposition given here is a complete combinatorial description of the embedding of $\Omega$ in $\mathbb{C}^2$.

## Chapter 2

# Programs and Projective Case

In the first chapter we provided an algorithmic method for constructing a regular CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ from the braid monodromy of $\Omega$, where $\Omega$ is an affine plane curve and $\mathcal{D}$ a large enough polydisc. We will present now a program in SageMath that implements this algorithm and that, for any curve, provides the decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ explicitly.

We have also written a second program, based on the first one, that provides a simplicial decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$. This decomposition is thin enough to take a regular neighborhood of the curve.

In the first two sections of this chapter we explain each of these programs. In the third section, which is essentially unrelated to the first two, we discuss briefly the problem of obtaining a CW decomposition of the pair $(\mathbb{P}^2, \Omega)$, for a given projective curve $\Omega$. We show how such a decomposition can be constructed, though we don't provide it explicitly as we did in the affine case.

Finally, since we will be working with the same objects, we keep all the definitions and notations of the previous chapter.

## 2.1 Program for a CW Decomposition

We begin by explaining the first program. Let us recall that the global decomposition for $(\mathcal{D}, \Omega \cap \mathcal{D})$ is constructed from the ordered sets $\{\{\beta^s_{(1)}, \ldots, \beta^s_{(l_s)}\}\}_{s=1}^m$ of sub-braids of the local braids, the conjugating braids $\alpha_1, \ldots, \alpha_m$, and factorizations for all of these. This program uses as its input the sets of braids for $\Omega$, and returns the regular CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ explicitly. The complete program can be found in appendix A.

It is worth noticing that, in [9], Carmona has given a program in Maple that calculates the braid monodromy of $\Omega$ from an equation for it. His program returns the braid monodromy in the form of a list of local braids and conjugating braids, and the local braids are separated, which means that the output of Carmona's program is exactly the input of ours. Therefore, Carmona's program together with ours allows to obtain a regular CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ from an equation for $\Omega$.

Now we describe the program. Within it, braids are represented in the following way. An Artin generator $\sigma_i^{\varepsilon}$ ($\varepsilon = \pm 1$) of any braid group is designated by the number $\varepsilon i$, and the identity $e$ of the same group by 0. A braid will be represented then by a list of the integers corresponding to a given factorization. Along this section we will refer freely to such lists as braids.

We start the program by defining the following classes.

- `LocalBraid`. An instance of this class will represent the ordered set of sub-braids $\beta_{(r)}$ of the local braid $\beta$ associated to a given $x_s \in \Delta$, along with the conjugating braid $\alpha$ associated to that same point. This class has the following fields. The field `sing_point`, containing the number $s$ of the critical value $x_s$. The field `braids`, containing the list of the $l$ sub-braids $\beta_{(r)}$. A field `n` containing the list $n_1, \ldots, n_l$, where $n_j$ is the number of strands $\beta_{(j)}$. And finally, a field `cong_braid` containing the conjugating braid $\alpha$.

- `BraidMonodromy`. This class has an only field `local_braids`, containing the list of the $m$ objects of type `LocalBraid` describing the $m$ elements of $\Delta$. An instance of this class contains therefore the complete information about the braid monodromy of a curve.

These two classes provide a structure for the input. We continue by providing a structure for the cell complex. This structure is based on the next three classes that we define.

- `Cell`. An instance of this class represents a cell.

- `CellWithSign`. An instance of this class represents a cell with sign. This class has two fields, one containing a number $\pm 1$ and the other one an object of type `Cell`.

- `Chain`. The instances of this class represent elements of the chain modules. It has an only field, `set_of_cells_w_sign`, that contains a set of objects of type `CellWithSign`.

Then we create classes for the different families of cells of our decomposition. Let us recall that our CW complex is formed by the union of the towers $T_{x_0}, \ldots, T_{x_m}$ and the bridges $B_1, \ldots, B_m$. To distinguish cells from different towers and bridges let us add to each cell the superindex $s$ to indicate to which $T_{x_s}$ or $B_s$ they belong to. Let us recall also that the set of cells of each tower $T_{x_s}$ is

$$\frac{\bigcup_{1\leq r\leq l_s}\left[\mathrm{Bnd}(H_r)\cup \mathrm{Con}(H_r)\cup\{AA_{(-,r)}\}\right]}{\phi_s}.$$

For each $T_{x_s}$ we define

$$\begin{aligned}\mathrm{Con}(T_{x_s}) \quad &: \quad = \bigcup\nolimits_{1\leq r\leq l_s}\mathrm{Con}(H_r) \text{ and}\\ \text{Non-Con}(T_{x_s}) \quad &: \quad = \frac{\bigcup_{1\leq r\leq l_s}\left[\mathrm{Bnd}(H_r)\bigcup\{AA_{(-,r)}\}\right]}{\phi_s}.\end{aligned}$$

To represent the cells in these sets we define the classes

- `ConeCell` and
- `TowerCell`,

respectively.

Let us consider the class `TowerCell` first. Each cell of Non-Con$(T_{x_s})$ is a function of several variables. A typical example would be

$$e^s_{j(i,r)},$$

which is function of its name "$e$" and the indexes $s$, $j$, $i$ and $r$. The meaning of these variables is explained below.

- Name: The name of the cell indicates its type or, more specifically, its place in the complex as we defined it.
- Index $j$: The index $0 \leq j \leq n_r$ (or $n_r + 1$ for some cells) is an integer indicating which strand of $\beta_{(r)}$ the cell is associated to. Some cells are not function of the set of strands and lack this index.
- Index $i$: The index $1 \leq i \leq k$ is an integer indicating which factor $\tau_r$ of $\beta_{(r)}$ the cell is associated to, or which $C_i$ it belongs to. Again, some cells lack this index. In those cases we write " $-$ " instead.
- Index $r$: The index $1 \leq r \leq l_s$ is an integer indicating which sub-braid $\beta_{(r)}$ of the local braid the cell is associated to, or which $H_r$ it belongs to. Once again, some cells lack this index, and in those cases we write " $-$ " instead.
- Index $s$: Finally, the index $0 \leq s \leq m$ indicates which critical value $x_s$ the cell is associated to, or which $T_{x_s}$ it belongs to.

The class `TowerCell` possesses fields for all of these variables, and two additional fields, one for the dimension of the cell, and one for the braid monodromy (object of type `BraidMonodromy`). These fields are called `dim`, `name`, `index` (for $j$), `sing_point`, `(i,r)`, and `mon`. As it can be seen, the two indexes $i$ and $r$ are included as a pair in a single field. The dimension and the braid monodromy are unnecessary from a mathematical point of view, but it is convenient to have them included. If a determined cell is lacking an index we fill the corresponding field with `none`. Thus, the cell $e^s_{j(i,r)}$ will be represented by the object

`TowerCell(1,e,j,s,(i,r),mon).`

It is important to notice that ghost cells admit to be represented in this way, even if they do not form part of the complex. This is, not only the cells Non-Con$(T_{x_s})$ but also the cells of $\bigcup_{1\le r\le l_s}\mathrm{Bnd}(H_r)$ can be represented as objects of this type. This is important because the program needs to do so in order to create conical cells and calculate their boundaries.

For this class we define a function `Border` that returns the boundary of any given cell in Non-Con$(T_{x_s})$ in the form of a set of objects of type `CellWithSign`. This function is calculated simply by the boundary formulae we have given.

Now let us consider the class `ConeCell`. Every cell $\vee\rho$ in $\mathrm{Con}(T_{x_s})$ has a base $\rho$ that belongs to $\mathrm{Bnd}(H_r)$, for some $r$, and might be a ghost cell. Therefore, the cells in $\mathrm{Con}(T_{x_s})$ are function of the cells in $\bigcup_{1\le r\le l_s}\mathrm{Bnd}(H_r)$, which always admit a representation as an object of type `TowerCell`. The class `ConeCell` has then a single field, called `cell`, containing an object of type `TowerCell` that represents the base of the conical cell.

As before, for this class we define a function `Border` that returns the boundary of any given cell in $\mathrm{Con}(T_{x_s})$ as a set of objects of type `CellWithSign`. This function is calculated simply by the boundary formulas

$$\begin{aligned}\partial_T(\vee\rho) &= AA_{(-,r)}-\rho \text{ and}\\ \partial_T(\vee\rho) &= (-1)^{\dim(\rho)}(-g(\rho))+\vee(\partial_{H_r}\rho).\end{aligned}$$

Now let us recall that the set of cells of each bridge $B_s$ is

$$\mathrm{Bot}(B_s)\cup\mathrm{Top}(B_s)\cup\mathrm{Prod}(B_s).$$

For these sets of cells we define the classes

- `BottomCell`,
- `TopCell` and
- `ProductCell`,

respectively.

Let us consider the class `BottomCell` first. The set $\mathrm{Bot}(B_s)$ is in turn composed by two disjoint subsets, one composed by the cells of the loom, and another one composed by the cells of the joints. A typical example of a cell on a joint would be

$$z^s_{r,j_r(i,0)},$$

whereas a typical cell on the loom would be

$$e^s_{j(i,0)}.$$

Since the cells on $\mathrm{Bot}(B_s)$ and $\mathrm{Top}(B_s)$ will be represented by different classes, the last subindex, taking the value 0 or 1, and used to distinguish cells from the top and the bottom, will become redundant and will be omitted from now on. The class `BottomCell` has fields for all the variables that these cells are dependent on, and additional fields for the dimension and the braid monodromy. These fields are called `dim`, `name`, `r`, `jr`, `sing_point`, `i` and `mon`. The field `r` will be used both for the subindex $r$ of the cells on the joints and the subindex $j$ of the cells on the loom. Thus, the cells $z^s_{r,j_r(i,0)}$ and $e^s_{j(i,0)}$ will be represented by the objects

`BottomCell(1,z,r,jr,s,i,mon)` and

`BottomCell(1,e,j,none,s,i,mon)`.

As usual, we define a function `Border`, calculated from the boundary formulae we have given.

Since the cells on $\mathrm{Top}(B_s)$ are an exact copy of those on $\mathrm{Bot}(B_s)$, the class `TopCell` is defined identically as `BottomCell`.

Finally, we define the class `ProductCell`. This class has two fields called `cellB` and `cellT` containing the objects of type `BottomCell` and `TopCell` that respectively represent the cells at $\mathrm{Bot}(B_s)$ and $\mathrm{Top}(B_s)$ corresponding to the product cell.

This class also has a function `border` calculated by the formulas

$$\begin{aligned}\partial(I\rho) &= \rho_1 - \rho_0, \\ \partial(I\rho) &= (-1)^{\dim(\rho)}(\rho_1 - \rho_0) + I(\partial(\rho)).\end{aligned}$$

It should be noticed that the product cells at the ends of bridge $B_s$, this is, the product cells on $D_0$ and $D_{kc+1}$, are identified with certain cells on $T_0$ and $T_{x_s}$. These identifications need to be taken into account. A function `product_of_chain is used` for this end. This function, when applied to a chain $c$, returns $I(c)$ but omitting the cells on $D_0$ and $D_{kc+1}$. The program uses this function as if it were the operator $I$ in the application of the formula $\partial(I\rho) = (-1)^{\dim(\rho)}(\rho_1 - \rho_0) + I(\partial(\rho))$. In this way, for any product cell $\rho$, the program calculates a chain that is the boundary of $\rho$ but omitting the cells on $D_0$ and $D_{kc+1}$. Then, the missing cells (i.e. those on on $T_0$ and $T_{x_s}$) are added explicitly to each boundary.

Once we have classes to represent all kind of cells, and functions to calculate its boundaries, all what is remaining to obtain a CW complex is to provide the list of cells. For this end we define the functions `cells_of_tower` and `cells_of_bridge` that return the cells of any given tower or bridge respectively, as a dictionary that assigns to each dimension the set of cells of that dimension.

A function `CW_decomposition`, defined for the class `BraidMonodromy`, uses these two functions to return all the cells of the CW complex, again as a dictionary.

Having the entire CW complex, it only remains to specify which cells belong to the curve $\Omega$. To this end we use a function called `in_curve`.

## 2.2 Program for a Simplicial Decomposition

We will explain now the second program. This program uses as its input a regular CW complex as returned by the first program, and returns a simplicial decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ explicitly. The complete program can be found in appendix B.

Let us recall that the output of the first program is a dictionary that assigns to each dimension a set of objects of type `Cell`, for which there are defined functions called `border`, and that this dictionary represents the CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$. Along this section we will refer freely to this kind of dictionaries as CW complexes.

Let $\alpha$ be a cell in the decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$, and $\beta$ a cell in $\partial\alpha$. Then there are objects $a$ and $b$ in $D$ that represent $\alpha$ and $\beta$. By calculating the border of $a$, the program creates a new object that is identical to $b$, but a different object nevertheless. In order to avoid this we create a new structure for the regular CW complexes.

We start by defining the following classes.

- `Simple_Cell`. An instance of this class represents a cell. This class has two fields, `dim` and `name`, that contain the dimension and name of the cell.
- `Cell_With_Sign`. An instance of this class represents a cell with sign. This class has two fields, one containing an object of type `Simple_Cell`, and the other one a number $\pm 1$.

We continue by defining a function `simple_complex`. This function uses a single parameter intended to be a CW complex. Let $D$ be a CW complex and, for every dimension $i$, let $X_i$ be the set assigned to it. Then, for each cell in $X_i$, the function creates an object of type `Simple_Cell` that has the name and dimension of the given cell, and then groups all the resulting objects in a set, that we will call $SX_i$. The function returns the list that assigns to each dimension $i$ the set $SX_i$ of objects of type `Simple_Cell`. Therefore, what the function `simple_complex` does is to take all the objects of type `Cell` within a CW complex, and replace them with equivalent objects of type `Simple_Cell`. Along this section we will refer freely to this kind of dictionaries as simple complexes.

Let $SD$ be a dictionary returned by `simple_complex`. Then, given $a$ and $b$ as before, there are objects $sa$ and $sb$ in $SD$ corresponding to $a$ and $b$. The function `simple_complex` also identifies *sb* with the object created by applying the function border to *sa*, thus avoiding the duplication issue explained before.

We also define a function `subcomplex` that tells which objects of type `Simple_Cell` in this dictionary represent cells on $\Omega$.

We will explain now a general algorithm to obtain, from a regular CW complex, a simplicial complex with the same underlying space. The algorithm in question is the one that runs from lower to higher dimension and transforms each cell into a star over its center. Although this algorithm is known and very simple, we explain it in order to make our program understandable.

Let $C$ be a CW complex of dimension $dim$, and let $C_i$ be the set of cells of $C$ of dimension $i$. Given a cell $\sigma$ on $C$, we denote the set of cells on the boundary of $\sigma$ by $\partial(\sigma)$. Finally, let $x$ be an element and $(y_1, \ldots, y_k)$ a $k$-tuple of elements. We define $x*(y_1, \ldots, y_k)$ as the $(k+1)$-tuple $(x, y_1, \ldots, y_k)$.

Let $K$ denote the simplicial complex we are about to build, and $K_i$ the set of simplices of $K$ of dimension $i$. We define $K_0$ as the set of one-tuples of elements of $C$.

In order to define the simplices of higher dimensions, for each cell $\sigma \in C$ and each dimension $0 \leq i \leq \dim(\sigma)$, we build certain sets that we will call $V_i(\sigma)$, $B_i(\sigma)$ and $W_i(\sigma)$. These sets represent the following:

$$V_i(\sigma) : \text{The } i\text{-simplices of } K \text{ lying on the interior of } \sigma.$$
$$B_i(\sigma) : \text{The } i\text{-simplices of } K \text{ lying on the boundary of } \sigma.$$
$$W_i(\sigma) : \text{The } i\text{-simplices of } K \text{ lying on the closure of } \sigma.$$

We build these sets by recursion in the following way. For each $\sigma \in C_0$ we define

$$\begin{aligned} V_0(\sigma) &= \emptyset, \\ B_0(\sigma) &= \{\sigma\}, \text{ and} \\ W_0(\sigma) &= \{\sigma\}. \end{aligned}$$

Now, let $1 \leq d \leq dim$ and let us assume that the sets $V_i(\sigma)$, $B_i(\sigma)$, and $W_i(\sigma)$ are already defined for the cells of $C_{d-1}$. Then, for every $\sigma \in C_0$ we define $V_i(\sigma)$, $B_i(\sigma)$ and $W_i(\sigma)$ as follows. For $i = 0$,

$$\begin{aligned} V_0(\sigma) &= \emptyset, \\ B_0(\sigma) &= \textstyle\bigcup_{\rho \in \partial(\sigma)} W_0(\rho), \text{ and} \\ W_0(\sigma) &= V_0(\sigma) \cup B_0(\sigma). \end{aligned}$$

And for $1 \leq i \leq d$,

$$\begin{aligned} V_i(\sigma) &= \{\sigma * \lambda \mid \lambda \in B_{i-1}(\sigma)\}, \\ B_i(\sigma) &= \textstyle\bigcup_{\rho \in \partial(\sigma)} W_i(\rho) \text{ (or } B_i(\sigma) = \emptyset \text{ if } i = d\text{), and} \\ W_i(\sigma) &= V_i(\sigma) \cup B_i(\sigma). \end{aligned}$$

Then we define

$$K = K_0 \cup \left( \bigcup_{\sigma \in C} \bigcup_{i=0}^{\dim(\sigma)} V_i(\sigma) \right),$$

which is a disjoint union.

It is easy to see that this algorithm, when applied to a regular CW complex, produces a simplicial complex with the same underlying space. And in particular, when applied to a simplicial complex, it produces its barycentric subdivision.

Let us recall that only regular CW complexes admit combinatorial descriptions. Specifically, a regular CW complex can be presented as a set of cells with border functions. Simplicial complexes are regular by definition and therefore can be presented as sets of tuples. This is the reason for which the algorithm is restricted to the regular case.

In fact, the fundamental idea of the algorithm also works for CW complexes that are not regular, though in this case it needs to be expressed in different terms, and the result is a CW complex in which every cell is the image of a simplex (and might not be regular). However, since the CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ we have designed is regular, we can follow the combinatorial approach. This approach has also the advantage that it allows to express the complexes by the relatively simple structures we have defined, while keeping all the nice properties of regular complexes.

Several variations of this algorithm can be useful. For example, for the purpose of transforming a CW complex into a simplicial complex, since one-dimensional cells are also one-dimensional simplices, a variation that omits the subdivision of the one-dimensional cells can also be used.

Let $S$ be a subcomplex of $C$. Another variation of the algorithm omits the subdivision of the one-dimensional cells, except by those that do not lie on $S$, but have both of its ends lying on $S$. This second variation is useful because it allows a regular neighborhood of $S$ to be taken on a barycentric subdivision of $K$.

The program continues with a function `from_CW_to_simplicial_with_sets` that is an implementation of the second variation. This function uses two parameters intended to be a simple complex and a subcomplex of this, and returns a dictionary that assigns to each dimension a set of tuples. These sets of tuples represent $K_1, \ldots, K_{dim}$ and the dictionary represents $K$. Along this section we will refer freely to this kind of dictionaries as simplicial complexes. A function `simplex_in_subcomplex` keeps track of the simplices produced by subdividing the subcomplex.

Another function, `subdivide`, is an implementation of the main subdivision algorithm we just described (with no variations). This function uses a single parameter, intended to be a regular CW or simplicial complex, and returns a simplicial complex. As we have already said, if used upon a simplicial complex, the complex returned is its barycentric subdivision. If `subdivide` is used upon the output of `from_CW_to_simplicial_with_sets`, a function `simplex_in_subcomplex` keeps track of the subsimplices coming from the original subcomplex.

By successively applying `simple_complex`, `from_CW_to_simplicial_with_sets` and `subdivide`, the program provides a simplicial complex, represented as a dictionary, that decomposes $(\mathcal{D}, \Omega \cap \mathcal{D})$.

The function `from_CW_to_simplicial_with_sets`, besides turning the CW complex into a simplicial complex, acts as a first barycentric subdivision with regard to $\Omega$. The function `subdivide` performs a barycentric subdivision itself. Therefore, the simplicial complex returned by the program is thin enough for taking a regular neighborhood of $\Omega$.

Two last functions, `reg_neig` and `comp_reg_neig`, return simplicial decompositions for a regular neighborhood of $\Omega$ and the complement of $\Omega$.

## 2.3 Decomposition for a Projective Plane Curve

Let $\Omega$ be a projective curve in $\mathbb{P}^2$. In this section we discuss how to obtain a CW decomposition of the pair $(\mathbb{P}^2, \Omega)$, as we did in the previous complement for the case of an affine curve.

Let $L_\infty$ be a line in $\mathbb{P}^2$ in generic position with regard to $\Omega$, this is, transversal to $\Omega$. Let $P$ be a point in $L_\infty$ not lying on $\Omega$. We can define coordinates in $\mathbb{P}^2 \backslash L_\infty$ in the following way. The pencil of lines through $P$ is parametrized by $\mathbb{P}^1$, and therefore, by removing $L_\infty$, the remaining lines of the pencil are parametrized by $\mathbb{C}$, providing a first coordinate $x$ (if we want the projection map on $x$ to be generic, we can also demand that $P$ does not belong to any non-generic line). A similar procedure, by taking another point of projection, provides a second coordinate $y$. In this way, we have a natural identification of $\mathbb{P}^2 \backslash L_\infty$ with $\mathbb{C}^2$.

Let $L_{y=0}$ be the line of $\mathbb{P}^2$ corresponding to the $x$ axis of $\mathbb{C}^2$. Then, on the pencil of lines through $P$, there are finitely many lines intersecting $\Omega$ non-transversally (tangent to $\Omega$ or passing through singularities). Let

$$\Delta = \{x_1, \ldots, x_m\}$$

be the set of these points. This is the same $\Delta$ defined in the previous chapter.

We will describe in the first place a CW complex decomposition for a regular neighborhood of the line at the infinity $L_\infty$. If we see $\mathbb{P}^2$ as a compactification of $\mathbb{C}^2$ in this way, then a closed regular neighorhood $R$ of $L_\infty$ is the complement of an open polydisc $D^\varepsilon \times D^\delta$. This implies that $\partial R$ is equal to $\partial(D^\varepsilon \times D^\delta)$ and homeomorphic to $S^3$. Then, $\partial R$ has a natural Heegaard splitting $\partial R = T_1 \cup_T T_2$, where

$$\begin{array}{rcl} T_1 & = & \partial D^\varepsilon \times D^\delta, \\ T_2 & = & D^\varepsilon \times \partial D^\delta \text{ and} \\ T & = & \partial D^\varepsilon \times \partial D^\delta. \end{array}$$

Let $D_1$ and $D_2$ be meridian disks for $T_1$ and $T_2$. Let $m$ and $l$ be the boundaries of $D_1$ and $D_2$ with given orientations.

Let us notice that if $L$ is any line of $\mathbb{P}^2$ passing through the origin, then $L \cap R$ is a disk centered at $L \cap L_\infty$. Let $p : R \longrightarrow L_\infty$ be the restriction to $R$ of the projection from the origin to the line at infinity. This map sends every disk $L \cap R$ onto its own center $L \cap L_\infty$, and endows $R$ with a fiber bundle structure, the base of which is $L_\infty$, and the fibers of which are the disks of the form $L \cap R$.

The restriction of $p$ to $\partial R$ is therefore a Hopf fibration on $\partial R$. Furthermore, it is easy to see that $T$ is a union of fibers of this Hopf fibration, which makes $T$ itself an $S^1$-fibered space. Let $h_1, \ldots, h_n$ be $n$ fibers of $T$. Then $h_1, \ldots, h_n$ are circumferences on $T$ homologous to $m + l$ (if $m$ and $l$ are given the right orientations), and form, with respect to $D_1$, a full twist braid inside $\partial R$. The situation is illustrated in the following figure.

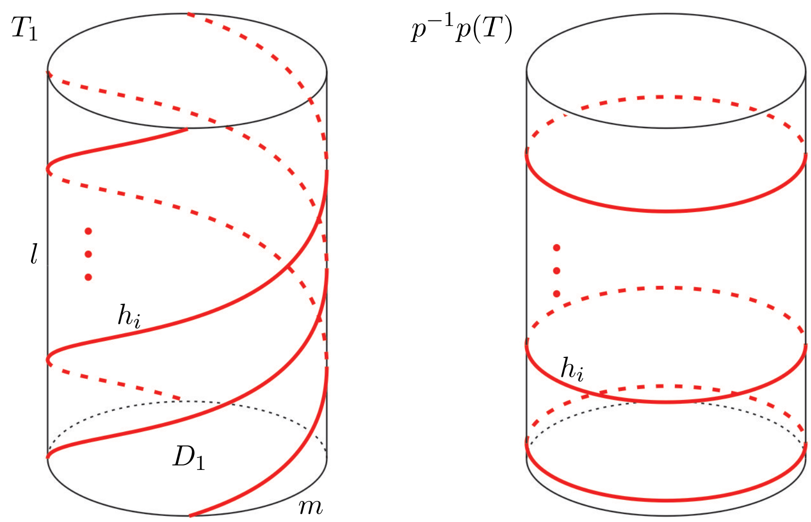


**Figure 2.1**

Then the one-skeleton

$$m \cup l \cup h_1 \cup \cdots \cup h_n$$

induces naturally a CW complex structure on $T$.

Let us consider now the torus $p^{-1}p(T)$ on $R$, which is bounded by $T$. Let us notice that $p^{-1}p(h_1), \ldots, p^{-1}p(h_n)$ is a set of fibers of $R$ that are meridian disks of $p^{-1}p(T)$. Then, we can endow $p^{-1}p(T)$ with the CW complex structure defined by the following one and two-skeletons:

$$\begin{array}{rcl} 1 & : & m \cup l \cup h_1 \cup \cdots \cup h_n \\ 2 & : & T \cup p^{-1}p(h_1) \cup \cdots \cup p^{-1}p(h_n). \end{array}$$

The CW complex structure of $T$ can be also extended to $T_1$ and $T_2$ as follows. We endow $T_1$ and $T_2$ with the CW complex structure defined by the following one and two-skeletons. For $T_1$:

$$
\begin{aligned}
1 &\;:\; m \cup l \cup h_1 \cup \cdots \cup h_n\,, \\
2 &\;:\; T \cup D_1.
\end{aligned}
$$

And for $T_2$:

$$
\begin{aligned}
1 &\;:\; m \cup l \cup h_1 \cup \cdots \cup h_n\,, \\
2 &\;:\; T \cup D_2.
\end{aligned}
$$

Let us observe that the three solid tori $T_1$, $T_2$ and $p^{-1}p(T)$ share $T$ as a common boundary. We have given these three solid tori CW complex decompositions all coincident on the common boundary $T$. Thus, we have a decomposition for $T_1 \cup_T T_2 \cup_T p^{-1}p(T)$.

Now, let $S_{Nth} := p(T_1)$, $S_{Sth} := p(T_2)$, and $e := p(T)$. Since $p\,|_{\partial R}$ is a Hopf fibration, then $S_{Nth}$ and $S_{Sth}$ are disks, and $e$ is their common boundary. The union of $S_{Nth}$ and $S_{Sth}$ equals $L_\infty$, so we can think of $e$ as an equator for $L_\infty$, and of $S_{Nth}$ and $S_{Sth}$ as northern and southern hemispheres.

Let $B_{Nth} := p^{-1}(S_{Nth})$ and $B_{Sth} := p^{-1}(S_{Sth})$. Then $B_{Nth}$ and $B_{Sth}$ are two four-dimensional balls, whose union is $R$ and whose interiors are disjoint. Let us notice that $B_{Nth}$ is the the product of $S_{Nth}$ and the fiber of $R$. This means that $B_{Nth}$ is the product of two disks, and therefore $\partial B_{Nth}$ has a natural Heegaard splitting, with solid tori $\bigcup_{x\in S_{Nth}} \partial p^{-1}(x)$ and $p^{-1}(\partial S_{Nth})$. We see that

$$
\begin{aligned}
\bigcup_{x\in S_{Nth}} \partial p^{-1}(x) &= \partial R \cap \left[\bigcup_{x\in S_{Nth}} p^{-1}(x)\right] = T_1\,, \\
p^{-1}(\partial S_{Nth}) &= p^{-1}(e) = p^{-1}p(T),
\end{aligned}
$$

and that the common boundary of these tori is $T$, which means that the Heegaard splitting for $\partial B_{Nth}$ is in fact

$$\partial B_{Nth} = T_1 \cup_T p^{-1}p(T).$$

Similarly, $\partial B_{Sth}$ has the natural Heegaard splitting

$$\partial B_{Sth} = T_2 \cup_T p^{-1}p(T).$$

This implies that the complement of the set $T_1 \cup_T T_2 \cup_T p^{-1}p(T)$ in $R$ is composed of two disjoint four-dimensional open balls. Since we already have a CW decomposition for $T_1 \cup_T T_2 \cup_T p^{-1}p(T)$, we have a decomposition for $R$.

We will discuss now how this decomposition, and the one defined in the previous chapter, induce a decomposition for $(\mathbb{P}^2, \Omega)$.

Let $D = D^\varepsilon \times D^\delta$ be a polydisc containing all the points of the form $(x_i, y) \in \Omega$ with $x_i \in \Delta$, and let $R$ be the complement of the interior of $D$ as before. If the line at the infinite $L_\infty$ is generic, as we have chosen it to be, then the intersection of $\Omega$ and $R$ consists of $n$ disks centered at $L_\infty$, the boundaries of which form a full twist on $\partial R$. By means of an isotopy, we can assume these disks are fibers of $R$, and that its boundaries lie on $T$.

By taking $h_1, \ldots, h_n$ as the boundaries of these disks (i.e., the $n$ components of $\Omega \cap \partial R$), and endowing $R$ with the CW complex structure just described, we obtain a CW decomposition for the pair $(R, R \cap \Omega)$, that we denote by $\tilde{R}$.

On the other hand, Theorem 1.18 provides us with a CW decomposition of the pair $(D, D \cap \Omega)$ that we denote by $\tilde{D}$. Let $\partial\tilde{R}$ and $\partial\tilde{D}$ be the CW complex structures induced by $\tilde{R}$ and $\tilde{D}$ on $\partial R = \partial D$. These structures do not coincide. However, $\partial\tilde{R}$ induce a subdivision of $\tilde{D}$ that we call $\tilde{D}'$, and $\partial\tilde{D}$ a subdivision of $\tilde{R}$, that we call $\tilde{R}'$.

Then $\tilde{R}'$ and $\tilde{D}'$ are CW decompositions of the pairs $(R, R \cap \Omega)$ and $(D, D \cap \Omega)$, respectively, that are coincident on $\partial R = \partial D$. Therefore, the union of $\tilde{R}'$ and $\tilde{D}'$ provide a CW decomposition of the pair $(\mathbb{P}^2, \Omega)$.

In the affine case we provided an explicit presentation of the decomposition of the pair $(D, D \cap \Omega)$. To provide the equivalent presentation for $(\mathbb{P}^2, \Omega)$ would be more difficult however, because $\partial\tilde{R}$ and $\partial\tilde{D}$ are quite different, and its intersection is hard to describe. A possible solution would be to separate $\partial R$ and $\partial D$ a small distance, leaving a space in between homeomorphic to $S^3 \times I$. This space could then be filled with a transitioning decomposition, as we did for the joints in the previous chapter.

Chapter 3

# A CW Decomposition of the Milnor Fiber of Singularities of the Form $z^n - x^a y^b$

In this chapter we study the topology of the compact Milnor fiber of the singularities of the form $f : (x, y, z) = z^n - x^a y^b$. Here, $f$ is a representative of a surface singularity germ $f : (\mathbb{C}^3, 0) \longrightarrow (\mathbb{C}, 0)$ and $a$, $b$, and $n$ are positive integers. Let $B_\varepsilon$ and $S_\varepsilon$ be the ball and sphere of radius $\varepsilon$ in $\mathbb{C}^3$ respectively.

Since $f$ is a quasi-homogeneous polynomial, it holds that, for every $\varepsilon > 0$, there exists a stratification of $f^{-1}(0)$ such that each stratum is transversal to $S_\varepsilon$. Therefore, the $\eta$ appearing in the definition of the Milnor fiber, as we presented it in the introduction, can be chosen to be arbitrarily large. In particular, for $\eta = 1$, and say $\varepsilon = 2$, we have that

$$f^{-1}(t) \pitchfork S_\varepsilon$$

for every $t$ with $0 < |t| \leq \eta$. Therefore, by taking $t = -1$, we obtain that the compact Milnor fiber of $f$ is given by the intersection of the surface

$$\mathcal{F} := \left\{(x, y, z) \in \mathbb{C}^3 \mid z^n - (x^a y^b - 1) = 0\right\}$$

with $B_\varepsilon$. We denote this compact Milnor fiber by $\mathcal{CF}$.

The purpose of this chapter is to construct a CW decomposition for $\mathcal{CF}$. To do this, we start by constructing a decomposition of a much simpler space, and then, through the use of coverings, we find decompositions for increasingly complicated spaces, until eventually reaching one for $\mathcal{CF}$.

## 3.1 Decomposition for a Hyperbola and its Asymptotes

We begin by finding a CW decomposition for a sufficiently large polydisc of $\mathbb{C}^2$ intersecting the set $\{(x,y) \in \mathbb{C}^2 \mid xy(xy-1)=0\}$ in a subcomplex. To this end we could use the method described in Chapter 1, however, in this particular case, we can build a much simpler decomposition. In this section we describe that decomposition.

As in the previous chapter, let us denote the complex lines in $\mathbb{C}^2$ by writing $L$ and the equation of the line as a subindex. Let us also define

$$\mathcal{H}_{1,1} := \Big\{(x,y) \in \mathbb{C}^2 \mid xy - 1 = 0\Big\},$$

which is a hyperbola. Let us observe that $\{(x,y) \in \mathbb{C}^2 \mid xy(xy-1)=0\} = \mathcal{H}_{1,1} \cup L_{x=0} \cup L_{y=0}$. Let

$$B := \Big\{(x,y) \in \mathbb{C}^2 \mid \|x\|, \|y\| \leq \varepsilon\Big\}$$

be large enough to ensure that $B \cap \mathcal{H}_{1,1}$ has a non-empty interior.

Now we will set the bases for our construction with some more definitions. For each $0 \leq \delta \leq \varepsilon$, let $S_\delta$ be the sphere defined by

$$S_\delta = \partial\Big\{(x,y) \in \mathbb{C}^2 \mid \|x\|, \|y\| \leq \delta\Big\} = \Big\{(x,y) \in \mathbb{C}^2 \mid \max\{\|x\|, \|y\|\} = \delta\Big\},$$

and let $T_{1,\delta}$, $T_{2,\delta}$ and $T_\delta$ be defined by

$$\begin{aligned} T_{1,\delta} &= \Big\{(x,y) \in \mathbb{C}^2 \mid \|x\| = \delta, \|y\| \leq \delta\Big\}, \\ T_{2,\delta} &= \Big\{(x,y) \in \mathbb{C}^2 \mid \|x\| \leq \delta, \|y\| = \delta\Big\} \text{ and} \\ T_\delta &= \Big\{(x,y) \in \mathbb{C}^2 \mid \|x\| = \|y\| = \delta\Big\}. \end{aligned}$$

Let us notice that, for each $\delta$, the sets $T_{1,\delta}$ and $T_{2,\delta}$ are two solid tori, with common boundary $T_\delta$, that constitute a Heegaard splitting for $S_\delta$. The situation is illustrated in Figure 3.1. Let us observe also that $\partial B = S_\varepsilon$.

We consider $B$ as having the conical structure $B = (S_\varepsilon \times [0,\varepsilon]) / (S_\varepsilon \times \{0\})$, defined by the following rule:

$$(p,t) := \frac{t}{\varepsilon}\, p \qquad \forall p \in S_\varepsilon, \forall t \in [0,\varepsilon].$$

Then, every three-dimensional fiber $S_\varepsilon \times \{\delta\}$ of $B$ is equal to $S_\delta$, and every one-dimensional fiber $\{p\} \times [0,\varepsilon]$ is equal to the segment $\overline{0p}$, which is a radius of $B$. Moreover, it holds for every $\delta$ that $T_{1,\varepsilon} \times \{\delta\} = T_{1,\delta}$ , $T_{2,\varepsilon} \times \{\delta\} = T_{1,\delta}$ and $T_\varepsilon \times \{\delta\} = T_\delta$, meaning that the Heegaard splittings $S_\delta = T_{1,\delta} \cup_{T_\delta} T_{2,\delta}$ are coherent with the conical structure of $B$.

For each $0 \leq \delta \leq \varepsilon$, let us define

$$\begin{aligned} co_{1,\delta} &= T_{1,\delta} \cap L_{y=0} = \left\{(x,0) \in \mathbb{C}^2 \mid \|x\| = \delta\right\}, \\ co_{2,\delta} &= T_{2,\delta} \cap L_{x=0} = \left\{(0,y) \in \mathbb{C}^2 \mid \|y\| = \delta\right\}, \\ m_\delta &= \left\{(x,y) \in \mathbb{C}^2 \mid x = \delta, \|y\| = \delta\right\}, \\ l_\delta &= \left\{(x,y) \in \mathbb{C}^2 \mid y = \delta, \|x\| = \delta\right\}. \end{aligned}$$

Then, $co_{1,\delta}$ and $co_{2,\delta}$ are cores for $T_{1,\delta}$ and $T_{2,\delta}$ respectively, while $m_\delta$ and $l_\delta$ are meridians for $T_{1,\delta}$ and $T_{2,\delta}$ respectively. We consider $co_{1,\delta}$, $co_{2,\delta}$, $m_\delta$ and $l_\delta$ with counterclockwise orientations in the spaces $L_{y=0}$, $L_{x=0}$, $L_{x=\delta}$ and $L_{y=\delta}$ respectively.

Let $m := \min\{\|(x,y)\| \mid (x,y) \in \mathcal{H}_{1,1}\}$ and $\Delta := \{(x,y) \in \mathcal{H}_{1,1} \mid \|(x,y)\| = m\}$. We will show that $\Delta$ lies within a single sphere $S_\delta$, that we call $S_{\delta_0}$. In fact, $\Delta$ is a circumference contained in $T_{\delta_0}$ and homologous to $-m_{\delta_0} + l_{\delta_0}$.

**Lemma 3.1.** *It holds that $\Delta = S_{\delta_0} \cap \mathcal{H}_{1,1}$ for some $\delta_0$. Moreover, $\Delta = -m_{\delta_0} + l_{\delta_0}$.*

*Proof.* Let $f : \mathbb{R}^+ \to \mathbb{R}$ be defined by $f(s) = \frac{s^2-1}{s}$. An analysis over the derivatives of $f$ shows that $f$ has its absolute minimum at $s = 1$, a fact that will be used later.

Let us notice that $\mathcal{H}_{1,1} = \left\{(z, z^{-1}) \mid z \in \mathbb{C}\right\}$. For every point $(z, z^{-1}) \in \mathcal{H}_{1,1}$, it holds that

$$\left\|(z, z^{-1})\right\|^2 = \|z\|^2 + \left\|z^{-1}\right\|^2 = \frac{\|z\|^4 + 1}{\|z\|^2} = f(\|z\|^2).$$

Therefore,

$$\begin{aligned} & \left\{z \mid \left\|(z, z^{-1})\right\| = m\right\} \\ = & \left\{z \mid \left\|(z, z^{-1})\right\|^2 \leq \left\|(w, w^{-1})\right\|^2 \ \forall w \in \mathbb{C}\right\} \\ = & \left\{z \mid f(\|z\|^2) \leq f(\|w\|^2) \ \forall w \in \mathbb{C}\right\} \\ = & \left\{z \mid \|z\|^2 = 1\right\} \\ = & \{e^{i\theta} \mid \theta \in \mathbb{R}\}. \end{aligned}$$

Which means that a given point $(z, z^{-1})$ belongs to $\Delta$ if and only if $f$ has an absolute minimum at $\|z\|^2$. Since we know that the absolute minimum of $f$ occurs at $s = 1$, then $(z, z^{-1})$ belongs to $\Delta$ if and only if $\|z\|^2 = 1$, that is, if and only if $z = e^{i\theta}$ for some $\theta \in \mathbb{R}$. Hence, we have that

$$\Delta = \left\{(z, z^{-1}) \mid \left\|(z, z^{-1})\right\| = m\right\} = \{(e^{i\theta}, e^{-i\theta}) \mid \theta \in \mathbb{R}\},$$

and this set is exactly $-m_1 + l_1$. By fixing $\delta_0 = 1$ we obtain the lemma. We have also obtained that $m = \sqrt{2}$. $\square$

From now on, we will denote $-m_\delta + l_\delta$ by $k_\delta$. On the other hand, for every $\delta_0 < \delta \leq \varepsilon$, let us notice that $S_\delta \cap \mathcal{H}_{1,1}$ has two connected components, one of them contained in $\mathring{T}_{1,\delta}$ and the other in $\mathring{T}_{2,\delta}$. Let $h_{1,\delta}$ and $h_{2,\delta}$ denote these components:

$$\begin{aligned} h_{1,\delta} &:= S_\delta \cap \mathcal{H}_{1,1} \cap T_{1,\delta} \ , \\ h_{2,\delta} &:= S_\delta \cap \mathcal{H}_{1,1} \cap T_{2,\delta} \ . \end{aligned}$$

Let us notice that $h_{1,\delta}$ and $h_{2,\delta}$ are circumferences ambient isotopic to $k_\delta$ in $S_\delta$. Moreover, $h_{1,\delta}$ and $h_{2,\delta}$ form a Hopf link inside $S_\delta$. If we allow $\delta$ to vary, we see that $h_{1,\delta}$ and $h_{2,\delta}$ tend both to $k_\delta$ as $\delta$ tends to $\delta_0$, and tend to $co_{1,\delta}$ and $co_{2,\delta}$ respectively as $\delta$ tends to infinity (if we allow $\delta$ to be greater than $\varepsilon$).

Then,

$$B \cap \mathcal{H}_{1,1} = k_0 \cup \bigcup\nolimits_{\delta_0 < \delta \leq \varepsilon} (h_{1,\delta} \cup h_{2,\delta}),$$

and this set is an annulus. The topology of the inclusion $B \cap \mathcal{H}_{1,1} \subset B$, and the objects we have defined are illustrated in Figure 3.1.

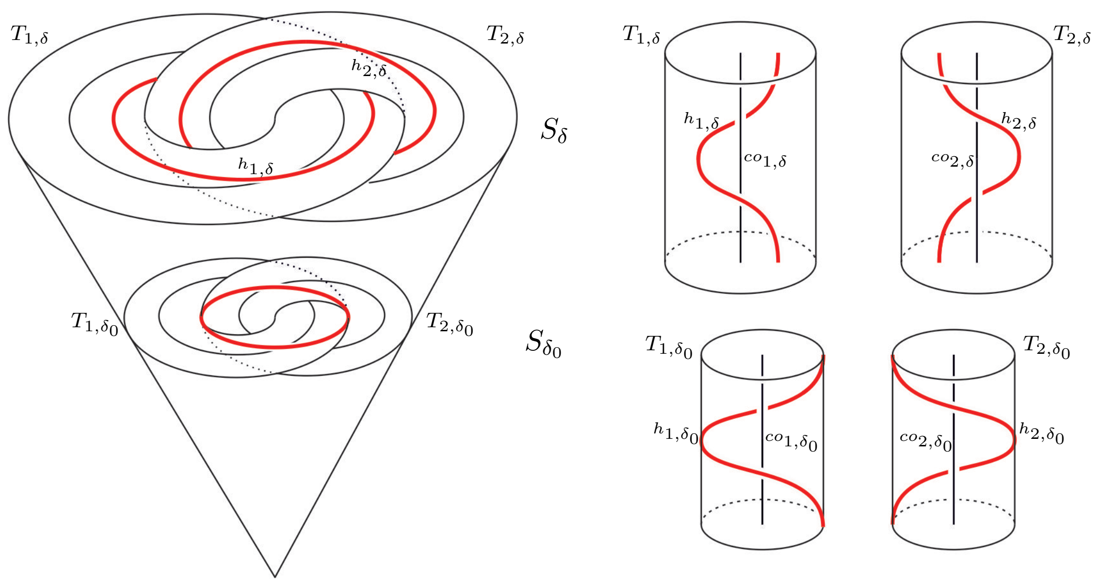


**Figure 3.1**

Let us proceed now with the construction of the CW decomposition. For every $\delta$ such that $\delta_0 \leq \delta \leq \varepsilon$, let us define

$$\begin{aligned} D_{1,\delta} &:= \left\{(x,y) \in \mathbb{C}^2 \mid x = \delta, \|y\| \leq \delta\right\} \cup \left\{(x,y) \in \mathbb{C}^2 \mid x \in [0,\delta], \|y\| = \delta\right\}, \\ D_{2,\delta} &:= \left\{(x,y) \in \mathbb{C}^2 \mid y = \delta, \|x\| \leq \delta\right\} \cup \left\{(x,y) \in \mathbb{C}^2 \mid y \in [0,\delta], \|x\| = \delta\right\}. \end{aligned}$$

The set $D_{1,\delta}$ is the union of the meridian disk of $T_{1,\delta}$ bounded by $m_\delta$, and the annulus contained in $T_{2,\delta}$ bounded by $m_\delta$ and $co_{2,\delta}$. Similarly, $D_{2,\delta}$ is the union of the meridian disk of $T_{2,\delta}$ bounded by $l_\delta$, and the annulus contained in $T_{1,\delta}$ bounded by $l_\delta$ and $co_{1,\delta}$.

Additionally, for every $\delta_0 \leq \delta \leq \varepsilon$ and $\theta \in \mathbb{R}$ let us define the segments

$$\begin{aligned} L_{1,\delta}(\theta) &:= \overline{(\delta e^{i\theta}, 0)(\delta e^{i\theta}, \delta e^{-i\theta})} \text{ and} \\ L_{2,\delta}(\theta) &:= \overline{(0, \delta e^{i\theta})(\delta e^{i\theta}, \delta e^{-i\theta})}. \end{aligned}$$

Let us observe that $(\delta e^{i\theta}, 0)$ is a point of $co_{1,\delta}$, $(0, \delta e^{i\theta})$ a point in $co_{2,\delta}$ and $(\delta e^{i\theta}, \delta e^{-i\theta})$ a point in $k_\delta$. Then, $\bigcup_{\theta\in\mathbb{R}} L_{1,\delta}(\theta)$ is an annulus contained in $T_{1,\delta}$, bounded by $co_{1,\delta}$ and $k_\delta$, and $\bigcup_{\theta\in\mathbb{R}} L_{2,\delta}(\theta)$ is an annulus contained in $T_{2,\delta}$, bounded by $co_{2,\delta}$ and $k_\delta$.

We may assume, by deforming $\mathcal{H}_{1,1}$, that for every $\delta_0 < \delta \leq \varepsilon$, $h_{1,\delta}$ is contained in $\bigcup_{\theta\in\mathbb{R}} L_{1,\delta}(\theta)$ and $h_{2,\delta}$ is contained in $\bigcup_{\theta\in\mathbb{R}} L_{2,\delta}(\theta)$. Let $A_{1,\delta}$ be the sub-annulus of $\bigcup_{\theta\in\mathbb{R}} L_{1,\delta}(\theta)$ bounded by $h_{1,\delta}$ and $k_\delta$, and $A_{2,\delta}$ that one of $\bigcup_{\theta\in\mathbb{R}} L_{2,\delta}(\theta)$ bounded by $h_{2,\delta}$ and $k_\delta$.

Now, for any fixed $\delta_0 < \delta \leq \varepsilon$, observe that the union $T_\delta \cup D_{1,\delta} \cup D_{2,\delta} \cup A_{1,\delta} \cup A_{2,\delta}$ is a two-dimensional CW complex having $h_{1,\delta} \cup h_{2,\delta}$ as a subcomplex, and whose complement in $S_\delta$ is composed by two three-dimensional open balls. Therefore, $T_\delta \cup D_{1,\delta} \cup D_{2,\delta} \cup A_{1,\delta} \cup A_{2,\delta}$ provide a CW complex structure for $(S_\delta, \mathcal{H}_{1,1} \cap S_\delta)$ that we will denote by $\mathcal{D}(S_\delta)$.

Similarly, For $\delta_0$, the union $T_{\delta_0} \cup D_{1,\delta_0} \cup D_{2,\delta_0}$ is a two-dimensional CW complex. To this complex we add $k_\delta$ as an edge, splitting $T_\delta$ into two cells, obtaining a two-dimensional complex of which $k_\delta$ is a subcomplex. As before, the complement of this complex in $S_{\delta_0}$ is composed by two three-dimensional open balls. Therefore, $T_{\delta_0} \cup D_{1,\delta_0} \cup D_{2,\delta_0} \cup k_{\delta_0}$ provide a CW complex structure for $(S_{\delta_0}, \mathcal{H}_{1,1} \cap S_{\delta_0})$ that we will denote by $\mathcal{D}(S_{\delta_0})$. After the previous discussion, the following lemma is clear.

**Lemma 3.2.** *The Complexes $\mathcal{D}(S_{\delta_0})$ and $\mathcal{D}(S_\delta)$ are well-defined CW decompositions for $(S_{\delta_0}, \mathcal{H}_{1,1} \cap S_{\delta_0})$ and $(S_\delta, \mathcal{H}_{1,1} \cap S_\delta)$ respectively.*

The complexes $\mathcal{D}(S_{\delta_0})$ and $\mathcal{D}(S_\varepsilon)$ are illustrated in Figures 3.3 and 3.2 respectively, with a name and orientation given to each cell. For convenience, we use different types of letters to denote cells according to the dimension: Uppercase Latin for dimension 0, lowercase Latin for dimension 1, lowercase Greek for dimension 2, uppercase Greek for dimension 3 and, again, uppercase Greek for dimension 4.

Let us define $S_{[\delta_0,\varepsilon]} := \bigcup_{\delta_0 \leq \delta \leq \varepsilon} S_\delta$. Our aim now will be to find a CW decomposition for this space, that we will call $\mathcal{D}(S_{[\delta_0,\varepsilon]})$. Let us consider an arbitrary cell $\rho_\varepsilon$ in $\mathcal{D}(S_\varepsilon)$. For every $\delta_0 < \delta \leq \varepsilon$, $\rho_\varepsilon$ has an equivalent cell $\rho_\delta$ in $\mathcal{D}(S_\delta)$, so we can define the set $\rho := \bigcup_{\delta_0 < \delta < \varepsilon} \rho_\delta$. Let us observe that, for every $\rho_\varepsilon$, $\rho$ is an open ball of dimension $\dim(\rho_\varepsilon) + 1$.

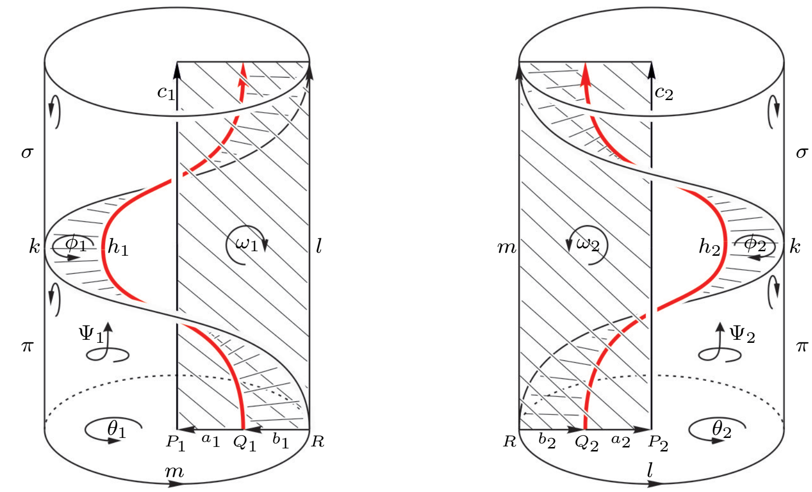


**Figure 3.2**

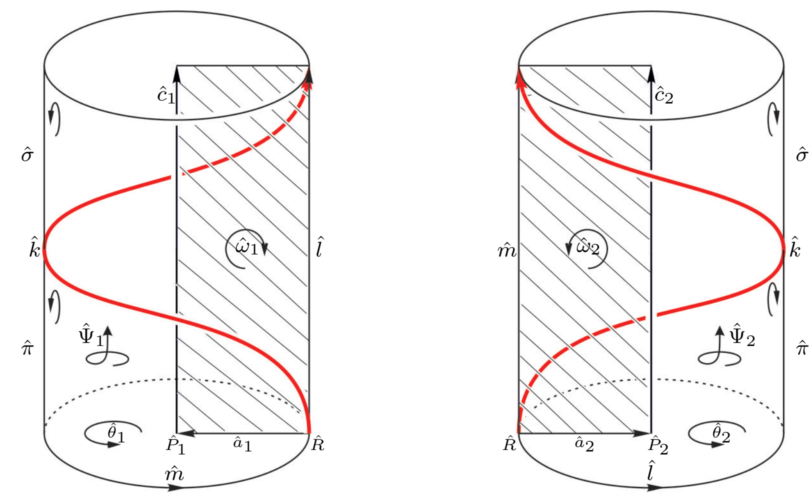


**Figure 3.3**

Let us denote the set of cells $\{\rho \mid \rho_\varepsilon \in \mathcal{D}(S_\varepsilon)\}$ by $\mathcal{D}(S_{(\delta_0,\varepsilon)})$. In the following table we assign a name to each cell in this set. The cells in $\mathcal{D}(S_\varepsilon)$ are listed in the first column by dimension, from 0 to 3. In the second column, in front of each cell $\rho_\varepsilon$, there is the name that we assign to the corresponding $\rho$ (the dimension of $\rho$ is greater that the dimension of $\rho_\varepsilon$ by one).

| Dim. 0 | Dim. 1 | $h_1$ | $\eta_1$ | $\omega_1$ | $\Omega_1$ |
|---|---|---|---|---|---|
| | | $h_2$ | $\eta_2$ | $\omega_2$ | $\Omega_2$ |
| $P_1$ | $p_1$ | $c_1$ | $\zeta_1$ | $\phi_1$ | $\Phi_1$ |
| $P_2$ | $p_2$ | $c_2$ | $\zeta_2$ | $\phi_2$ | $\Phi_2$ |
| $Q_1$ | $q_1$ | $a_1$ | $\alpha_1$ | $\sigma$ | $\Sigma$ |
| $Q_2$ | $q_2$ | $a_2$ | $\alpha_2$ | $\pi$ | $\Pi$ |
| $R$ | $r$ | $b_1$ | $\beta_1$ | | |
| | | $b_2$ | $\beta_2$ | Dim. 3 | Dim. 4 |
| Dim. 1 | Dim. 2 | | | | |
| | | Dim. 2 | Dim. 3 | $\Psi_1$ | $\Xi_1$ |
| $m$ | $\mu$ | | | $\Psi_2$ | $\Xi_2$ |
| $l$ | $\lambda$ | $\theta_1$ | $\Theta_1$ | | |
| $k$ | $\kappa$ | $\theta_2$ | $\Theta_2$ | | |

We define $\mathcal{D}(S_{[\delta_0,\varepsilon]})$ by

$$\mathcal{D}(S_{[\delta_0,\varepsilon]}) = \mathcal{D}(S_{\delta_0}) \cup \mathcal{D}(S_{(\delta_0,\varepsilon)}) \cup \mathcal{D}(S_\varepsilon)\,.$$

The definition of $\rho$ ensures that this is a well defined CW complex. We refer to the cells of $\mathcal{D}(S_\varepsilon)$, $\mathcal{D}(S_{(\delta_0,\varepsilon)})$ and $\mathcal{D}(S_{\delta_0})$ as the *upper*, *middle* and *lower* cells of $\mathcal{D}(S_{[\delta_0,\varepsilon]})$ respectively. The boundaries of all the cells of $\mathcal{D}(S_{[\delta_0,\varepsilon]})$ are given below.

Dimension 1:

*Upper*

$$\begin{array}{ll}
\partial(m) = R - R & \\
\partial(l) = R - R & \\
\partial(k) = R - R & \\
\partial(h_1) = Q_1 - Q_1 & \partial(h_2) = Q_2 - Q_2 \\
\partial(c_1) = P_1 - P_1 & \partial(c_2) = P_2 - P_2 \\
\partial(a_1) = P_1 - Q_1 & \partial(a_2) = P_2 - Q_2 \\
\partial(b_1) = Q_1 - R & \partial(b_2) = Q_2 - R
\end{array}$$

*Middle*

$$\begin{array}{ll}
\partial(p_1) = P_1 - \hat{P}_1 & \partial(p_2) = P_2 - \hat{P}_2 \\
\partial(q_1) = Q_1 - \hat{R} & \partial(q_2) = Q_2 - \hat{R} \\
\partial(r) = R - \hat{R} &
\end{array}$$

*Lower*

$$\begin{array}{lll}
\partial(\hat{m}) = \hat{R} - \hat{R} & \partial(\hat{c}_1) = \hat{P}_1 - \hat{P}_1 & \partial(\hat{c}_2) = \hat{P}_2 - \hat{P}_2 \\
\partial(\hat{l}) = \hat{R} - \hat{R} & \partial(\hat{a}_1) = \hat{P}_1 - \hat{R} & \partial(\hat{a}_2) = \hat{P}_2 - \hat{R} \\
\partial(\hat{k}) = \hat{R} - \hat{R} & &
\end{array}$$

Dimension 2:

*Upper*

$$\begin{array}{ll}
\partial(\sigma) = l - m - k & \\
\partial(\pi) = m - k - l & \\
\partial(\theta_1) = m + a_1 + b_1 - a_1 - b_1 & \partial(\theta_2) = l + a_2 + b_2 - a_2 - b_2 \\
\partial(\omega_1) = c_1 - l + a_1 + b_1 - a_1 - b_1 & \partial(\omega_2) = c_2 - m + a_2 + b_2 - a_2 - b_2 \\
\partial(\phi_1) = h_1 - k + b_1 - b_1 & \partial(\phi_2) = h_2 + k + b_2 - b_2
\end{array}$$

*Middle*

$$\begin{array}{ll}
\partial(\mu) = r - r + \hat{m} - m & \\
\partial(\lambda) = r - r + \hat{l} - l & \\
\partial(\kappa) = r - r + \hat{k} - k & \\
\partial(\eta_1) = q_1 - q_1 + \hat{k} - h_1 & \partial(\eta_2) = q_2 - q_2 - \hat{k} - h_2 \\
\partial(\zeta_1) = p_1 - p_1 + \hat{c}_1 - c_1 & \partial(\zeta_2) = p_2 - p_2 + \hat{c}_2 - c_2 \\
\partial(\alpha_1) = p_1 - q_1 + \hat{a}_1 - a_1 & \partial(\alpha_2) = p_2 - q_2 + \hat{a}_2 - a_2 \\
\partial(\beta_1) = q_1 - r - b_1 & \partial(\beta_2) = q_2 - r - b_2
\end{array}$$

*Lower*

$$\begin{array}{ll}
\partial(\hat{\sigma}) = \hat{l} - \hat{m} - \hat{k} & \\
\partial(\hat{\pi}) = \hat{m} + \hat{k} - \hat{l} & \\
\partial(\hat{\theta}_1) = \hat{m} + \hat{a}_1 - \hat{a}_1 & \partial(\hat{\theta}_2) = \hat{l} + \hat{a}_2 - \hat{a}_2 \\
\partial(\hat{\omega}_1) = \hat{c}_1 - \hat{l} + \hat{a}_1 - \hat{a}_1 & \partial(\hat{\omega}_2) = \hat{c}_2 - \hat{m} + \hat{a}_2 - \hat{a}_2
\end{array}$$

Dimension 3:

*Upper*

$$\begin{array}{l}
\partial(\Psi_1) = \sigma + \pi + \theta_1 - \theta_1 + \omega_1 - \omega_1 + \phi_1 - \phi_1 \\
\partial(\Psi_2) = -\sigma - \pi + \theta_2 - \theta_2 + \omega_2 - \omega_2 + \phi_2 - \phi_2
\end{array}$$

*Middle*

$$\begin{aligned}
\partial(\Theta_1) &= \mu + \alpha_1 + \beta_1 - \alpha_1 - \beta_1 + \theta_1 - \hat{\theta}_1 \\
\partial(\Theta_2) &= \lambda + \alpha_2 + \beta_2 - \alpha_2 - \beta_2 + \theta_2 - \hat{\theta}_2 \\
\partial(\Omega_1) &= \zeta_1 - \lambda + \alpha_1 + \beta_1 - \alpha_1 - \beta_1 + \omega_1 - \hat{\omega}_1 \\
\partial(\Omega_2) &= \zeta_2 - \mu + \alpha_2 + \beta_2 - \alpha_2 - \beta_2 + \omega_2 - \hat{\omega}_2 \\
\partial(\Phi_1) &= \eta_1 - \kappa + \beta_1 - \beta_1 + \phi_1 \\
\partial(\Phi_2) &= \eta_2 + \kappa + \beta_2 - \beta_2 + \phi_2 \\
\partial(\Sigma) &= \lambda - \mu - \kappa + \sigma - \hat{\sigma} \\
\partial(\Pi) &= \mu + \kappa - \lambda + \pi - \hat{\pi}
\end{aligned}$$

*Lower*

$$\begin{aligned}
\partial(\hat{\Psi}_1) &= \hat{\sigma} + \hat{\pi} + \hat{\theta}_1 - \hat{\theta}_1 + \hat{\omega}_1 - \hat{\omega}_1 \\
\partial(\hat{\Psi}_2) &= -\hat{\sigma} - \hat{\pi} + \hat{\theta}_2 - \hat{\theta}_2 + \hat{\omega}_2 - \hat{\omega}_2
\end{aligned}$$

Dimension 4:

$$\begin{aligned}
\partial(\Xi_1) &= \Sigma + \Pi + \Theta_1 - \Theta_1 + \Omega_1 - \Omega_1 + \Phi_1 - \Phi_1 - \Psi_1 + \hat{\Psi}_1 \\
\partial(\Xi_2) &= -\Sigma - \Pi + \Theta_2 - \Theta_2 + \Omega_2 - \Omega_2 + \Phi_2 - \Phi_2 - \Psi_2 + \hat{\Psi}_2
\end{aligned}$$

Now let us define $S_{[0,\delta_0]} := \bigcup_{0\le\delta\le\delta_0} S_\delta$. As we did with $S_{[\delta_0,\varepsilon]}$, we will now find a convenient CW decomposition for $S_{[0,\delta_0]}$. Let us notice that, by extending $\mathcal{D}(S_{\delta_0})$ conically to the origin, we can readily obtain a CW decomposition for $S_{[0,\delta_0]}$. From this decomposition and $\mathcal{D}(S_{[\delta_0,\varepsilon]})$ we obtain a CW decomposition for $B$, satisfying our initial requirement that $B \cap \mathcal{H}_{1,1}$, $B \cap L_{x=0}$ and $B \cap L_{y=0}$ are all subcomplexes. However, we will not use this complex in the successive constructions because it abounds in cells that provide no essential information. We will instead deviate a little from our original purpose, allowing the decomposition of $B$ not to meet entirely the coordinate axes in a subcomplex.

Let $\Upsilon := int(S_{[0,\delta_0]}) = \bigcup_{0\le\delta<\delta_0} S_\delta$, which is an open four-dimensional ball. Then we define the CW decomposition $\mathcal{D}(B)$ for $B$ by

$$\mathcal{D}(B) = \{\Upsilon\} \cup \mathcal{D}(S_{[\delta_0,\varepsilon]}).$$

**Theorem 3.3.** The complex $\mathcal{D}(B)$ is a well defined CW decomposition for $B$. The intersections $B \cap \mathcal{H}_{1,1}$, $S_{[\delta_0,\varepsilon]} \cap L_{x=0}$ and $S_{[\delta_0,\varepsilon]} \cap L_{y=0}$ are subcomplexes of $\mathcal{D}(B)$.

*Proof.* To see that $\mathcal{D}(B)$ is well defined it suffices to observe that $\partial\Upsilon = S_{\delta_0}$ is a subcomplex of $\mathcal{D}(S_{[\delta_0,\varepsilon]})$. All $B \cap \mathcal{H}_{1,1} = S_{[\delta_0,\varepsilon]} \cap \mathcal{H}_{1,1}$, $S_{[\delta_0,\varepsilon]} \cap L_{x=0}$ and $S_{[\delta_0,\varepsilon]} \cap L_{y=0}$ are subcomplexes by construction. □

## 3.2 Decomposition for the Curve $x^a y^b - 1 = 0$

Let $a$ and $b$ be positive integers. We will describe now a CW decomposition for a sufficiently large polydisc of $\mathbb{C}^2$ intersecting the set $\mathcal{H}_{a,b} := \left\{(x, y) \in \mathbb{C}^2 \mid x^a y^b - 1 = 0\right\}$ in a subcomplex. We construct such CW decomposition by lifting the complexes we already have through the use of branched coverings.

Let us define

$$B' := \left\{(x, y) \in \mathbb{C}^2 \mid \|x\| \leq \sqrt[a]{\varepsilon}, \|y\| \leq \sqrt[b]{\varepsilon}\right\}.$$

We will soon see that $B'$ intersects $\mathcal{H}_{a,b}$ in a space in which every connected component has non-empty interior, for which it is a sufficiently large polydisc as desired. This is the polydisc we will decompose.

Now let $P$ be the following map:

$$\begin{aligned} P : \mathbb{C}^2 &\to \mathbb{C}^2 \\ (x, y) &\mapsto (x^a, y^b) \end{aligned} .$$

This map is an $ab$-fold covering of $\mathbb{C}^2$ over $\mathbb{C}^2$, branched over one or two of the coordinate axes (except for the trivial case $a = b = 1$). In fact, it is the composition of the two cyclic coverings $(x, y) \mapsto (x^a, y)$ and $(x, y) \mapsto (x, y^b)$. The crucial fact that makes $P$ important for us is that

*The map $P|_{\mathcal{H}_{a,b}}$ is an unbranched covering of $\mathcal{H}_{a,b}$ over $\mathcal{H}_{1,1}$.*

Also,

*The map $P|_{B'}$ is a branched covering of $B'$ over $B$.*

Making this last statement true was the motivation to define $B'$ the way we did. Besides, it implies that $B'$ intersects $\mathcal{H}_{a,b}$ in the desired way. The branching set of $P|_{B'}$ is $B \cap L_{x=0}$ if $a > 1$ and $b = 1$, $B \cap L_{y=0}$ if $a = 1$ and $b > 1$, and $B \cap (L_{x=0} \cup L_{y=0})$ if $a > 1$ and $b > 1$. The trivial case $a = b = 1$ results in an empty branching set and is not of interest to us, since it is the case of the previous section.

Now we use $P$ to construct a CW complex $\mathcal{D}_{a,b}(B')$ for $(B', B' \cap \mathcal{H}_{a,b})$. We define this complex by

$$\mathcal{D}_{a,b}(B') := \left\{P^{-1}(\varsigma) \mid \varsigma \in \mathcal{D}(B)\right\}.$$

**Theorem 3.4.** The complex $\mathcal{D}_{a,b}(B')$ is a well defined CW decomposition for $(B', B' \cap \mathcal{H}_{a,b})$.

*Proof.* Let us notice that the branching set of the covering

$$P|_{P^{-1}(S_{[\delta_0,\varepsilon]})} : P^{-1}(S_{[\delta_0,\varepsilon]}) \longrightarrow S_{[\delta_0,\varepsilon]}$$

is either $S_{[\delta_0,\varepsilon]} \cap L_{x=0}$, $S_{[\delta_0,\varepsilon]} \cap L_{y=0}$, or the union of both sets. In any case, this branching set is a subcomplex of $\mathcal{D}(S_{[\delta_0,\varepsilon]})$, which implies that the complex $\left\{P^{-1}(\varsigma) \mid \varsigma \in \mathcal{D}(S_{[\delta_0,\varepsilon]})\right\}$ is well defined.

On the other hand, the fact that $\delta_0 = 1$, implies that $P^{-1}(S_{\delta_0}) = S_{\delta_0}$ and $P^{-1}(\Upsilon) = \Upsilon$. Then we have the following.

- $P^{-1}(S_{\delta_0})$ is a subcomplex of $\left\{P^{-1}(\varsigma) \mid \varsigma \in \mathcal{D}(S_{[\delta_0,\varepsilon]})\right\}$.
- $P^{-1}(\Upsilon)$ is a cell.
- $\partial P^{-1}(\Upsilon) = P^{-1}(S_{\delta_0})$

This implies that

$$\left\{P^{-1}(\Upsilon)\right\} \cup \left\{P^{-1}(\varsigma) \mid \varsigma \in \mathcal{D}(S_{[\delta_0,\varepsilon]})\right\}$$

is a well defined CW complex, and this complex is by definition $\mathcal{D}_{a,b}(B')$.

Besides, it holds that $B' \cap \mathcal{H}_{a,b} = P^{-1}(B \cap \mathcal{H}_{1,1})$. Since $B \cap \mathcal{H}_{1,1}$ is a subcomplex of $\mathcal{D}(B)$, $B' \cap \mathcal{H}_{a,b}$ is a subcomplex of $\mathcal{D}_{a,b}(B')$. □

It is worth noticing that this good definition, as well as some following lemmas, rely on the construction of our CW complexes. Actually, we have designed these complexes purposely in such a way that they include the branching and splitting complexes as subcomplexes, thus making these statements true.

## 3.3 Topology of the Curve $x^a y^b - 1 = 0$

Now that we have the desired decomposition $\mathcal{D}_{a,b}(B')$, we will describe the topology of the inclusion $B' \cap \mathcal{H}_{a,b} \subset B'$ and the combinatorics of $\mathcal{D}_{a,b}(B')$. The topology of the inclusion $B' \cap \mathcal{H}_{a,b} \subset B'$ can be inferred from that of $B \cap \mathcal{H}_{1,1} \subset B$ by means of $P$.

In order to do this we must first find a splitting complex for $P$. In this context, a splitting complex means a three-dimensional sumcomplex of $\mathcal{D}(B)$, of which the branching set of $P$ is a subcomplex, and the complement of which is simply connected. The general definition and main properties of splitting complexes can be found in [36].

Let $F_P$ denote the cone of $D_{1,\varepsilon} \cup D_{2,\varepsilon}$ in $B$.

**Lemma 3.5.** *The set $F_P$ is a splitting complex for $P$, and $F_P \backslash \Upsilon$ is a subcomplex of $\mathcal{D}(B)$.*

*Proof.* That $F_P \backslash \Upsilon$ is a subcomplex of $\mathcal{D}(B)$ is clear from the construction of $\mathcal{D}(B)$. The cells composing $F_P \backslash \Upsilon$ are $\Theta_1$, $\Theta_2$, $\Omega_1$, $\Omega_2$ and all the cells in $\partial(\Theta_1)$, $\partial(\Theta_2)$, $\partial(\Omega_1)$, $\partial(\Omega_2)$.

To see that $F_P$ is a splitting complex for $B$ it is enough to observe that the complement of $D_{1,\varepsilon} \cup D_{2,\varepsilon}$ in $\partial B \backslash (L_{x=0} \cup L_{y=0}) = \partial B \backslash \{co_{1,\varepsilon} \cup co_{2,\varepsilon}\}$, is simply connected. This property is preserved by the conical structure. □

Then, $P$ allows us to construct $B'$ from copies of $B$ by using elementary covering theory as follows. We consider $ab$ copies of $B$ and denote them by $\{B_{i,j}\}_{i\in[[1,a]],\ j\in[[1,b]]}$, where $[[\cdot,\cdot]]$ denotes closed intervals in $\mathbb{Z}$. We cut each of these copies along $F_P$, i.e. along the cones of $D_{1,\varepsilon}$ and $D_{2,\varepsilon}$. Foevery $i \in [[1,a]]$ and $j \in [[1,b]]$ let the sets $X_i$ and $Y_j$ be defined by $X_j := \{B_{1,j}, \dots, B_{a,j}\}$ and $Y_i := \{B_{i,1}, \dots B_{i,b}\}$. Then, on each $X_i$ we glue the $a$ copies of $B$ cyclically along $D_{1,\varepsilon}$, and on each $Y_i$ we glue the $b$ copies cyclically along $D_{2,\varepsilon}$. The space resulting from this gluing is $B'$, and each copy of $B$ is projected into $B$ by $P$.

Furthermore, we can make this gluing in such a way that each copy of $B\backslash\Upsilon$ is projected into $B\backslash\Upsilon$, and each copy of $\Upsilon$ into $\Upsilon$. In other words, we can construct $B'\backslash\Upsilon$ by gluing the copies of $B\backslash\Upsilon$ (on each $B_{i,j}$), and construct $\Upsilon \subset B'$ by gluing the copies of $\Upsilon$ (on each $B_{i,j}$). In the case of $B'\backslash\Upsilon$, since $F_P\backslash\Upsilon$ is a subcomplex of $\mathcal{D}(B)$, this gluing can be made cellularly, i.e. by identifying cells with cells. This is enough to affirm the following.

**Lemma 3.6.** *Each copy $B_{i,j}\backslash\Upsilon$ of $B\backslash\Upsilon$ intersects $B'\backslash\Upsilon$ in a subcomplex of $\mathcal{D}_{a,b}(B')\backslash\{\Upsilon\}$, and this subcomplex is a copy of $\mathcal{D}(B)\backslash\{\Upsilon\}$, cut along $F_P\backslash\Upsilon$.*

From these constructions it can be seen that $\mathcal{H}_{a,b}$ is something that could be described as a multiple hyperbola, made of $ab$ copies of $\mathcal{H}_{1,1}$ glued together. The following lemma implies that the number of connected components of $\mathcal{H}_{a,b}$ is $\gcd(a,b)$. Each of these components resemble $\mathcal{H}_{1,1}$ in the fact that they are made of one curve close to the origin from which two wings open to the infinite. Let $c := \gcd(a,b)$ and $m' := \min\{\|z\| \mid z \in \mathcal{H}_{a,b}\}$. Then we have the following.

**Lemma 3.7.** *For the set of points of $\mathcal{H}_{a,b}$ with minimum magnitude the following identity holds:*

$$\{z \in \mathcal{H}_{a,b} \mid \|z\| = m'\} = S_{\delta_0} \cap \mathcal{H}_{a,b} = P^{-1}(k_{\delta_0})\,.$$

*Moreover, this set is the disjoint union of $c$ curves homologous to $\frac{a}{c}m_{\delta_0} + \frac{b}{c}l_{\delta_0}$ on $T_{\delta_0}$.*

*Proof.* Since $\|(x,y)\| < \|(z,w)\| \Longrightarrow \|P(x,y)\| < \|P(z,w)\|$ for every $(x,y),(z,w) \in \mathbb{C}^2$, it holds that

$$\left\{z \in P^{-1}(\mathcal{H}_{1,1}) \mid \|z\| = m'\right\} = P^{-1}\left(\left\{z \in \mathcal{H}_{1,1} \mid \|z\| = \sqrt{2}\right\}\right).$$

Therefore

$$\{z \in \mathcal{H}_{a,b} \mid \|z\| = m'\} = P^{-1}(k_{\delta_0}).$$

Now let us observe that for every $(x,y),(z,w) \in \mathbb{C}^2$, with $P(x,y) = (z,w)$, it holds that $\|x\| = \|y\| = 1$ if and only if $\|z\| = \|w\| = 1$. Hence, since $k_{\delta_0} \subset T_{\delta_0}$, then $\{z \in \mathcal{H}_{a,b} \mid \|z\| = m'\} \subset T_{\delta_0}$. Furthermore, Because all the points of $T_{\delta_0}$ have equal magnitude (in fact, equal to $\sqrt{2}$), it holds that $\{z \in \mathcal{H}_{a,b} \mid \|z\| = m'\} = T_{\delta_0} \cap \mathcal{H}_{a,b}$. And since all the points of $S_{\delta_0}\backslash T_{\delta_0}$ have magnitude strictly lesser than $\sqrt{2}$, this last equality implies that

$$\{z \in \mathcal{H}_{a,b} \mid \|z\| = m'\} = S_{\delta_0} \cap \mathcal{H}_{a,b},$$

and $m' = m = \sqrt{2}$.

Let us now examine the set $P^{-1}(k_{\delta_0})$. We know that $k_{\delta_0} = \Big\{(e^{i\theta}, e^{-i\theta}) \mid \theta \in \mathbb{R}\Big\}$. Then $P^{-1}(k_{\delta_0}) = \Big\{(\sqrt[a]{e^{i\theta}}, \sqrt[b]{e^{-i\theta}}) \mid \theta \in \mathbb{R}\Big\}$ and therefore

$$\begin{aligned} P^{-1}(k_{\delta_0}) &= \Big\{(e^{i(\frac{\theta}{a}+k\frac{2\pi}{a})}, e^{-i(\frac{\theta}{b}+j\frac{2\pi}{b})}) \mid \theta \in \mathbb{R},\, k \in [[0,a]],\, j \in [[0,b]]\Big\} \\ &= \bigcup\nolimits_{k\in[[0,a]],\, j\in[[0,b]]} \Big\{(e^{i(\frac{\theta}{a}+k\frac{2\pi}{a})}, e^{-i(\frac{\theta}{b}+j\frac{2\pi}{b})}) \mid \theta \in \mathbb{R}\Big\}. \end{aligned}$$

For every $k \in [[0,a]]$ and $j \in [[0,b]]$ let us define

$$\begin{aligned} c_{k,j} &: = \Big\{(e^{i(\frac{\theta}{a}+k\frac{2\pi}{a})}, e^{-i(\frac{\theta}{b}+j\frac{2\pi}{b})}) \mid \theta \in \mathbb{R}\Big\}, \\ X &: = \{c_{k,j} \mid k \in [[0,a]],\, j \in [[0,b]]\}. \end{aligned}$$

Then,

$$P^{-1}(k_{\delta_0}) = \bigcup_{c_{k,j}\in X} c_{k,j}.$$

Let us notice that for any given $k$ and $j$, $c_{k,j}$ is a curve in $T_{\delta_0}$. However, several of these curves may be coincident; i.e., there may be $d, e \in [[0,a]]$ and $f, g \in [[0,a]]$ such that $c_{d,f} = c_{e,g}$.

Let us define

$$\begin{aligned} \varphi : \mathbb{Z}_a \oplus \mathbb{Z}_b &\twoheadrightarrow X \\ (k,j) &\mapsto c_{k,j} \end{aligned},$$

which is a surjective function; and let $N = \langle(1,1)\rangle \trianglelefteq \mathbb{Z}_a \oplus \mathbb{Z}_b$, where $\langle\cdot\rangle$ denotes generated by. It follows from the equations that, for every $k$ and every $j$, $c_{k,j} = c_{k+1,j+1}$. Then, for every $(k,j) \in \mathbb{Z}_a \oplus \mathbb{Z}_b$ and every $(d,e) \in N$ we have that $\varphi((d,e)+(k,j)) = \varphi(k,j)$. As a result,

$$\varphi' : \tfrac{\mathbb{Z}_a\oplus\mathbb{Z}_b}{N} \twoheadrightarrow X$$

$$N + (k,j) \mapsto c_{k,j}$$

is well defined and surjective.

We know that $\frac{\mathbb{Z}_a\oplus\mathbb{Z}_b}{N}$ is isomorphic to $\mathbb{Z}_c$ and in fact $\frac{\mathbb{Z}_a\oplus\mathbb{Z}_b}{N} = \{(0,0),(0,1),\ldots,(0,c)\}$. Then, the surjectivity of $\varphi'$ implies that

$$\begin{aligned} X &= \{\varphi(0,0), \varphi(0,1), \ldots, \varphi(0,c)\} \\ &= \{c_{0,0}, \ldots, c_{0,c}\}. \end{aligned}$$

It is easy to see from the equations of $c_{0,0}, \ldots, c_{0,c}$ that these are all different curves, even disjoint. Then $X$ is a set of $c$ elements and

$$\begin{aligned} P^{-1}(k_{\delta_0}) &= c_{0,0} \cup \cdots \cup c_{0,c} \\ &= \bigcup\nolimits_{j\in[[0,c]]} \Big\{(e^{i\frac{\theta}{a}}, e^{-i(\frac{\theta}{b}+j\frac{2\pi}{b})}) \mid \theta \in \mathbb{R}\Big\}. \end{aligned}$$

All these curves are homologous to $\frac{a}{c} m_{\delta_0} + \frac{b}{c} l_{\delta_0}$, which gives us the result. $\square$

Now we can present a simple topological model for $(B', B' \cap \mathcal{H}_{a,b})$. Let us define

$$\begin{aligned} T'_\varepsilon &: = \left\{ (x,y) \in \mathbb{C}^2 \mid \|x\| = \sqrt[a]{\varepsilon}, \|y\| = \sqrt[b]{\varepsilon} \right\}, \\ m'_\varepsilon &: = \left\{ (x,y) \in \mathbb{C}^2 \mid x = \sqrt[a]{\varepsilon}, \|y\| = \sqrt[b]{\varepsilon} \right\}, \\ l'_\varepsilon &: = \left\{ (x,y) \in \mathbb{C}^2 \mid y = \sqrt[b]{\varepsilon}, \|x\| = \sqrt[a]{\varepsilon} \right\}. \end{aligned}$$

And let $F_Q$ be the union of $c$ disjoint solid tori in $B'$, each of them intersecting $\partial B'$ in an annulus contained in $T'_\varepsilon$ with core homologous to $\frac{a}{c} m'_\varepsilon + \frac{b}{c} l'_\varepsilon$ in $T'_\varepsilon$. Let $A_\partial$ be the union of those $c$ annuli, and $A := \partial F_Q \backslash int(A_\partial)$. Then we have the following topological characterization of $(B', B' \cap \mathcal{H}_{a,b})$.

**Theorem 3.8.** The pairs $(B', B' \cap \mathcal{H}_{a,b})$ and $(B', A)$ are homeomorphic.

*Proof.* We will see that $B' \cap \mathcal{H}_{a,b}$ and $A$ are ambient isotopic in $B'$. Let us first examine the case for $a = b = 1$. In this case $B'$ is $B$, $F_Q$ is a single solid torus, $A$ is an annulus, and $B' \cap \mathcal{H}_{a,b} = B \cap \mathcal{H}_{1,1}$ is also an annulus, as we showed in the previous section. Let $H : (B \cap \mathcal{H}_{1,1}) \times I \longrightarrow B$ be an isotopy, constant on $h_{1,\varepsilon} \cup h_{2,\varepsilon}$, and pushing $B \cap \mathcal{H}_{1,1}$ into an annulus $\mathcal{H}_\partial$ contained in $\partial B$. Then, since the cores of $\mathcal{H}_\partial$ and $A_\partial$ are homologous (both homologous to $m'_\varepsilon + l'_\varepsilon$), we may deform $B$ in order that $\mathcal{H}_\partial$ coincides with $A_\partial$. Subsequently, we may deform $B$ in order that $B \cap \mathcal{H}_{1,1}$ coincides with $A$.

We can reason in a similar way for the general case. In this case, as a direct consequence of the previous lemma and the definition of $F_Q$, the $2c$ components of $\partial A_\partial$ may be forced to coincide with $2c$ annuli obtained by pushing $B' \cap \mathcal{H}_{a,b}$ into $\partial B$. $\square$

Actually, the sets $A$ and $F_Q$ could have been constructed on an arbitrary closed four-dimensional ball.

## 3.4 Combinatorics of $\mathcal{D}_{a,b}(B')$

Now we are interested in describing the combinatorics of $\mathcal{D}_{a,b}(B')$. Let us observe that given any cell of $\varsigma \in \mathcal{D}(B)$, the preimage $P^{-1}(\varsigma)$ consists of a disjoint union of cells which are copies of the original $\varsigma$, and which we call *the preimages* under $P$ of $\varsigma$. It is a consequence of Lemmas 3.5 and 3.6 that, given any $\varsigma \in \mathcal{D}(B) \backslash \{\Upsilon\}$, every copy $B_{i,j}$ of $B$ contains exactly one preimage of $\varsigma$, though several copies of $B$ may share the same one. For every $\varsigma \in \mathcal{D}(B) \backslash \{\Upsilon\}$ we choose a single preimage of $\varsigma$, which we denote by $\tilde{\varsigma}$. We may further choose all these preimages $\tilde{\varsigma}$ inside a single privileged copy of $B$, which we may assume is $B_{1,1}$. By adding $\Upsilon$ to this set of chosen preimages we obtain a subset $\check{\mathcal{B}}$ of $\mathcal{D}_{a,b}(B')$ that we will call the *first copy complex* of $\mathcal{D}_{a,b}(B')$, which is a set of cells but not a well defined CW complex. Notice that $\check{\mathcal{B}}$ contains exactly one preimage of each $\varsigma \in \mathcal{D}(B)$. We will be able to observe later that, despite the underlying space of the first copy complex is not $B_{1,1}$, it is almost as if it were because $\Upsilon$, which is the only odd cell, behaves conveniently similar to the origin.

Let $\check{t}, \check{s} : \mathbb{C}^2 \longrightarrow \mathbb{C}^2$ be defined by

$$\begin{aligned} \check{t}(x,y) &= (e^{\frac{2\pi}{a}} x, y) \text{ and} \\ \check{s}(x,y) &= (x, e^{\frac{2\pi}{b}} y). \end{aligned}$$

Then $\check{t}$ and $\check{s}$ are generators of the deck transformations group of the coverings $(x,y) \mapsto (x^a, y)$ and $(x,y) \mapsto (x, y^b)$ respectively. Together, the two generate the deck transformations group of $P$. Thus, when applied to $B'$, $\check{t}$ and $\check{s}$ cyclically permute the $ab$ copies of $B$ of which $B'$ is made. Moreover these copies are all the translations of $B_{1,1}$ by $\check{t}$ and $\check{s}$.

Let us notice also that for every cell $\rho$ in $\mathcal{D}_{a,b}(B')$, the images $\check{t}(\rho)$ and $\check{s}(\rho)$ are also cells in $\mathcal{D}_{a,b}(B')$. In consequence, $\check{t}$ and $\check{s}$ define functions from $\mathcal{D}_{a,b}(B')$ to $\mathcal{D}_{a,b}(B')$ that we keep calling $\check{t}$ and $\check{s}$. We can extend these functions linearly to the chain groups of $\mathcal{D}_{a,b}(B')$ as follows:

Let $C_i$ denote the chain group of $\mathcal{D}_{a,b}(B')$ of dimension $i$. Then, for every $0 \leq i \leq 4$ we define $\check{t}, \check{s} : C_i \longrightarrow C_i$ by

$$\begin{aligned} \check{t}(\rho_1 + \cdots + \rho_j) :=& \quad \check{t}(\rho_1) + \cdots + \check{t}(\rho_j) \text{ and} \\ \check{s}(\rho_1 + \cdots + \rho_j) :=& \quad \check{s}(\rho_1) + \cdots + \check{s}(\rho_j), \end{aligned}$$

which are in fact homomorphisms. For simplicity, we will often write $\check{t}\rho$ and $\check{s}\rho$ instead of $\check{t}(\rho)$ and $\check{s}(\rho)$.

Let us observe that given a cell $\varsigma \in \mathcal{D}(B) \backslash \{\Upsilon\}$, the application of $\check{t}$ and $\check{s}$ cyclically permute the preimages of $\varsigma$ among the copies of $B$ that make $B'$. Moreover, the preimages of $\varsigma$ are exactly all the translations of $\tilde{\varsigma}$ by $\check{t}$ and $\check{s}$.

Since $L_{x=0}$ is the branching set of $(x,y) \mapsto (x^a, y)$, for every $\varsigma \in \mathcal{D}(B) \backslash \{\Upsilon\}$ lying on $L_{x=0}$, it holds that $\check{t}$ acts trivially over the preimages of $\varsigma$. Similarly, for every $\varsigma \in \mathcal{D}(B) \backslash \{\Upsilon\}$ lying on $L_{y=0}$, it holds that $\check{s}$ acts trivially over the preimages of $\varsigma$. Finally, regarding $\Upsilon$, both $\check{t}$ and $\check{s}$ act trivially over it due to its symmetry, i.e. $\check{t}(\Upsilon) = \check{s}(\Upsilon) = \Upsilon$. For no other cell of $\mathcal{D}_{a,b}(B')$ do $\check{t}$ or $\check{s}$ act trivially. Thus, we have the following.

**Lemma 3.9.** *The cells of $\mathcal{D}_{a,b}(B')$ are exactly the following, and satisfy the properties described.*

- *The cell $\Upsilon$. Both $\check{t}$ and $\check{s}$ act trivially over $\Upsilon$.*
- *The preimages of $P_1$, $c_1$, $p_1$, $\zeta_1$, $\hat{P}_1$ and $\hat{c}_1$, which are the six cells of $\mathcal{D}(B) \backslash \{\Upsilon\}$ lying on $L_{x=0}$. If $\varsigma$ is one of these cells, its preimages are $\tilde{\varsigma}$, $\check{s}(\tilde{\varsigma})$,..., $\check{s}^{b-1}(\tilde{\varsigma})$. Only $\check{t}$ acts trivially over these cells.*
- *The preimages of $P_2$, $c_2$, $p_2$, $\zeta_2$, $\hat{P}_2$ and $\hat{c}_2$, which are the six cells of $\mathcal{D}(B) \backslash \{\Upsilon\}$ lying on $L_{y=0}$. If $\varsigma$ is one of these cells, its preimages are $\tilde{\varsigma}$, $\check{t}(\tilde{\varsigma})$,..., $\check{t}^{b-1}(\tilde{\varsigma})$. Only $\check{s}$ acts trivially over these cells.*

- *The preimages of all the remaining cells of $\mathcal{D}(B)\backslash\{\Upsilon\}$. If $\varsigma$ is one of these cells, its preimages are $\left\{\check{t}^{\,i}\check{s}^{\,j}(\tilde{\varsigma})\right\}_{0\leq i\leq a-1,\ 0\leq j\leq b-1}$. Neither $\check{t}$ or $\check{s}$ act trivially over these cells.*

Now we want to calculate the boundaries of all the cells of $\mathcal{D}_{a,b}(B')$. The following lemma is a consequence of the definition of $\mathcal{D}_{a,b}(B')$, $\check{t}$ and $\check{s}$.

**Lemma 3.10.** *For every $\varsigma \in \mathcal{D}_{a,b}(B')$, and every $i, j \in \mathbb{Z}$,*

$$\partial(\check{t}^{\,i}\check{s}^{\,j}\varsigma) = \check{t}^{\,i}\check{s}^{\,j}\partial(\varsigma)\,.$$

Lemma 3.9 implies that, once the boundaries of the cells of $\check{\mathcal{B}}$ have been calculated, this formula provides us the boundaries of all the cells of $\mathcal{D}_{a,b}(B')$. The boundaries of the cells of $\check{\mathcal{B}}$ are given at the end of the chapter (assuming $u = 1$).

## 3.5 Decomposition of the Milnor Fiber

Let $a$, $b$, and $n$ be positive integers. We will describe now a CW decomposition for the intersection of the surface

$$\mathcal{F} := \left\{(x, y, z) \in \mathbb{C}^3 \mid z^n - (x^a y^b - 1) = 0\right\}$$

with a sufficiently large polydisc.

Let $\mathcal{CF} := \mathcal{F} \cap (B' \times \mathbb{C})$. Let us observe that $\mathcal{CF}$ is bounded, since for any $(x, y, z) \in B' \times \mathbb{C}$ with $z^n - (x^a y^b - 1) = 0$ it holds that $\|z\|^n = \left\|x^a y^b - 1\right\| \leq \left\|x^a y^b\right\| + 1 \leq \varepsilon^2 + 1$. In fact, this bound is optimal, which implies that $\mathcal{CF}$ is also closed and, in consequence, compact. In fact, this is the compact Milnor fiber of $f$ we have already defined, and the set we will decompose.

Now let $Q$ be the following map:

$$\begin{aligned} Q : \mathcal{F} &\to \mathbb{C}^2 \\ (x, y, z) &\mapsto (x, y) \end{aligned}\,.$$

It is easy to confirm that

*The map $Q$ is a cyclic $n$-fold covering of $\mathcal{F}$ over $\mathbb{C}^2$ branched along $\mathcal{H}_{a,b}$.*

Furthermore,

*The map $Q\mid_{\mathcal{CF}}$ is a cyclic $n$-fold covering of $\mathcal{CF}$ over $B'$ branched along $B' \cap \mathcal{H}_{a,b}$.*

Now we use $Q$ to construct a CW complex $\mathcal{D}(\mathcal{CF})$ for $\mathcal{CF}$. We define this complex by

$$\mathcal{D}(\mathcal{CF}) := \left\{Q^{-1}(\varsigma) \mid \varsigma \in \mathcal{D}_{a,b}(B')\right\}.$$

**Theorem 3.11.** The complex $\mathcal{D}(\mathcal{CF})$ is a well defined CW decomposition for $\mathcal{CF}$.

*Proof.* This is true because the branching set $B' \cap \mathcal{H}_{a,b}$ of $Q$ is a subcomplex of $\mathcal{D}_{a,b}(B')$. □

As before, this good definition, as well as some of the following lemmas, rely on the construction of our CW complexes. We have designed these complexes purposely in such a way that they include the branching and splitting complexes of both $P$ and $Q$ as subcomplexes, which is the crucial fact that bound these statements to be true and allows us advance in our constructions.

## 3.6 Topology of the Milnor Fiber

Up to this point we have defined the space $\mathcal{CF}$ and the CW decomposition $\mathcal{D}(\mathcal{CF})$. However, we do not have a topological description of $\mathcal{CF}$, nor any combinatorial information about $\mathcal{D}(\mathcal{CF})$. We will now give a complete topological description of $\mathcal{CF}$. In order to do this we must find a splitting complex $F_Q$ for $Q$.

Let $\tau$ denote the solid torus $\bar{\Phi}_1 \cup \bar{\Phi}_2$ in $B$, and $F_Q$ the set $P^{-1}(\tau)$ in $B'$. Let us notice that this definition of $F_Q$ coincides with the one we gave in Section 3.3 (up to isotopy).

**Lemma 3.12.** *The set $F_Q$ is both a subcomplex of $\mathcal{D}_{a,b}(B')$ and a splitting complex for $Q$.*

*Proof.* To see that $F_Q$ is a subcomplex, it is enough to observe that $\tau$ is a subcomplex of $B$. The cells composing $\tau$ are the upper cells $Q_1$, $Q_2$, $R$, $b_1$, $b_2$, $h_1$, $h_2$, $k$, $\phi_1$ and $\phi_2$; the middle cells $q_1$, $q_2$, $r$, $\beta_1$, $\beta_2$, $\eta_1$, $\eta_2$, $\kappa$, $\Phi_1$ and $\Phi_2$; and the lower cells $\hat{R}$ and $\hat{\kappa}$ (as defined in Section 3.1, p. 54, 55).

On the other hand, to see that $F_Q$ is a splitting complex for $Q$ we need to prove that $(B' \backslash \mathcal{H}_{a,b}) \backslash F_Q$ is simply connected, which is the same as showing that $B' \backslash F_Q$ is simply connected.

We will show first that $B \backslash \tau$ is simply connected. Let us observe that $\bar{\phi}_1 \cup \bar{\phi}_2$ is an annulus contained in $\partial\tau$, with core homologous in $\tau$ to the core of $\tau$. As a consequence, $B \backslash \tau$ and $B \backslash (\bar{\phi}_1 \cup \bar{\phi}_2)$ are homeomorphic. To prove that $B \backslash (\bar{\phi}_1 \cup \bar{\phi}_2)$ is simply connected we first observe that $(\bar{\phi}_1 \cup \bar{\phi}_2) \subset \partial B$ and that $int(B \backslash (\bar{\phi}_1 \cup \bar{\phi}_2)) = \mathring{B}$ is a ball. Let $\gamma$ be a loop on $B \backslash (\bar{\phi}_1 \cup \bar{\phi}_2)$ with some base point at $\mathring{B}$. If $\gamma$ is contained in $\mathring{B}$ then it is trivial. On the contrary, if $\gamma$ intersects $\partial B$, then $\gamma$ may be slightly deformed to submerge it into $\mathring{B}$, for which $\gamma$ is homotopic to some loop on $\mathring{B}$, and therefore trivial.

We conclude that $B \backslash \tau$ is simply connected. A similar argument shows that $B' \backslash F_Q$ is simply connected. □

Then, $Q$ allows us to construct $\mathcal{CF}$ by using elementary covering theory as follows. We consider $n$ copies of $B'$ and denote them by $\{B'_k\}_{k \in [[1,n]]}$, Then we cut them along $F_Q$ and glue them cyclically along this cutting. The space resulting from this gluing is $\mathcal{CF}$, and every $B'_k$ is projected into $B'$ by $Q$. Thus, we have the following.

**Theorem 3.13.** Let $M$ be the 4-manifold obtained by cutting $n$ copies of $B'$ along $F_Q$, and gluing them cyclically along this cutting. Then $M$ is homeomorphic to $\mathcal{CF}$.

Let us recall that $F_Q$ was defined in Section 3.3 in purely topological terms as a certain union of tori contained in the four-dimensional ball. Therefore, we have obtained a purely topological definition of a manifold $M$ that is homeomorphic to the compact Milnor fiber of $f$ or, in other words, a topological characterization for this fiber.

Furthermore, since $F_Q$ is a subcomplex of $\mathcal{D}(\mathcal{CF})$, this gluing can be made cellularly. This implies that the space $\mathcal{CF}$ obtained by the gluing possesses the CW decomposition resulting from the lifting of $\mathcal{D}_{a,b}(B')$, which is $\mathcal{D}(\mathcal{CF})$ by definition. Then we have the following.

**Lemma 3.14.** *Each copy $B'_k$ of $B'$ intersects $\mathcal{CF}$ in a subcomplex of $\mathcal{D}(\mathcal{CF})$, and this subcomplex is a copy of $\mathcal{D}_{a,b}(B')$, cut along $F_Q$.*

We can think then that the gluing process not only produces $\mathcal{CF}$ from $B'$, but also $\mathcal{D}(\mathcal{CF})$ from $\mathcal{D}_{a,b}(B')$.

## 3.7 Combinatorics of $\mathcal{D}(\mathcal{CF})$

Now we are interested in describing the combinatorics of $\mathcal{D}(\mathcal{CF})$. We shall think about the composite covering $P \circ Q : \mathcal{CF} \longrightarrow B$. The branching set of this covering is

$$(B \cap L_{x=0}) \cup (B \cap L_{y=0}) \cup P(B' \cap \mathcal{H}_{a,b})$$

$$= B \cap [L_{x=0} \cup L_{y=0} \cup \mathcal{H}_{1,1}],$$

with the omission of $L_{x=0}$ if $a = 1$ and $L_{y=0}$ if $b = 1$. The splitting complex is

$$F_P \cup P(F_Q) = F_P \cup \tau.$$

Let us recall that $\mathcal{CF}$ is made by gluing the $n$ spaces $\{B'_k\}_{k \in [[1,n]]}$. Since each $B'_k$ is a copy of $B'$, each $B'_k$ is made from $ab$ copies of $B$, that we denote by $\{B_{i,j,k}\}_{i \in [[1,a]],\ j \in [[1,b]]}$. Then $\mathcal{CF}$ is made from $abn$ copies of $B$, cut along $F_P \cup \tau$ and then glued together in the way indicated by $P$ and $Q$. It is easy to observe that each copy $B_{i,j,k}$ is projected into $B$ by $P \circ Q$. Besides, since $(F_P \cup \tau) \backslash \Upsilon$ is a subcomplex of $\mathcal{D}(B)$, most of this gluing can be made cellularly. Thus we have the following.

**Lemma 3.15.** *For every cell $\rho \in \mathcal{D}_{a,b}(B')$, the preimage $Q^{-1}(\rho)$ consists of a disjoint union of cells which are copies of $\rho$. Every $B'_k$ contains exactly one of these preimages, though several copies may share the same one.*

From here it easily follows that for every $\varsigma \in \mathcal{D}(B)$, the preimage $(P \circ Q)^{-1}(\varsigma)$ consists of a disjoint union of cells which are copies of the original $\varsigma$. And that if $\varsigma \neq \Upsilon$, every $B_{i,j,k}$ contains exactly one preimage under $P \circ Q$ of $\varsigma$, though several copies may share the same one.

Now we choose a single privileged space $B'_k$, let us say $B'_1$, and let us observe that $B_{1,1,1}$ is just the copy of $B_{1,1}$ inside $B'_1$. The space $B'_1$ contains also a copy of the first copy complex $\check{\mathcal{B}}$, which we call $\mathcal{B}$. It is easily seen that given any $\varsigma \in \mathcal{D}(B)$, $\mathcal{B}$ contains exactly one preimage $\varsigma$ under $P \circ Q$, which we will denote by $\varsigma'$. The argument for this is the following: Let $\varsigma \in \mathcal{D}(B)$, and recall that $\check{\mathcal{B}}$ is a subset of $\mathcal{D}_{a,b}(B')$ that contains exactly one preimage $\tilde{\varsigma}$ of $\varsigma$ under $P$. By the lemma, $B'_1$ contains exactly one preimage of $\tilde{\varsigma}$ under $Q$. Then, by definition of $\mathcal{B}$, we obtain that $\mathcal{B}$ contains exactly one preimage of $\varsigma$ under $P \circ Q$.

Now, let $t, s, u : \mathbb{C}^3 \longrightarrow \mathbb{C}^3$ be defined by

$$\begin{aligned} t(x,y,z) &= (e^{\frac{2\pi}{a}}x, y, z), \\ s(x,y,z) &= (x, e^{\frac{2\pi}{b}}y, z) \text{ and} \\ u(x,y,z) &= (x, y, e^{\frac{2\pi}{n}}z). \end{aligned}$$

Let us show that $t$, $s$ and $u$ are deck transformations of $P \circ Q$. In the case of $t$, it holds that $Q \circ t = \check{t} \circ Q$ and, consequently, $P \circ Q \circ t = P \circ \check{t} \circ Q = P \circ Q$, which means that $t$ is a deck transformation of $P \circ Q$. The same argument applies for $s$. Besides, $t$ and $s$ act over each $B'_k$ exactly as $\check{t}$ and $\check{s}$ do over $B'$.

On the other hand, $u$ is a deck transformation of $Q$, which makes it a deck transformation of $P \circ Q$. Moreover, $u$ generates the deck transformation group of $Q$ and, when applied to $\mathcal{CF}$, $u$ permute cyclically the $n$ spaces $\{B'_k\}_{k \in [[1,n]]}$.

In fact, $t$, $s$ and $u$ generate the deck transformations group of $P \circ Q$. The three transformations $t$, $s$ and $u$ permute cyclically the elements of $\{B_{i,j,k}\}_{i \in [[1,a]],\ j \in [[1,b]], k \in [[1,n]]}$ in the variables $i$, $j$ and $k$ respectively, which implies that the spaces $\{B_{i,j,k}\}_{i \in [[1,a]],\ j \in [[1,b]], k \in [[1,n]]}$ that compose $\mathcal{CF}$ are all the translations of $B_{1,1,1}$ under $t$, $s$ and $u$.

Now let us notice that for every cell $\rho$ in $\mathcal{D}(\mathcal{CF})$ the images $t(\rho)$, $s(\rho)$ and $u(\rho)$ are also cells in $\mathcal{D}(\mathcal{CF})$. Hence, $t$, $s$ and $u$ define functions from $\mathcal{D}(\mathcal{CF})$ to $\mathcal{D}(\mathcal{CF})$ that we keep calling $t$, $s$ and $u$. We can extend these functions linearly to the chain groups of $\mathcal{D}(\mathcal{CF})$ as follows: Let $C_i$ denote the chain group of $\mathcal{D}(\mathcal{CF})$ of dimension $i$. Then, for every $0 \leq i \leq 4$ we define $t, s, u : C_i \longrightarrow C_i$ by

$$\begin{aligned} t(\rho_1 + \cdots + \rho_j) &: = t(\rho_1) + \cdots + t(\rho_j), \\ s(\rho_1 + \cdots + \rho_j) &: = s(\rho_1) + \cdots + s(\rho_j) \text{ and} \\ u(\rho_1 + \cdots + \rho_j) &: = u(\rho_1) + \cdots + u(\rho_j), \end{aligned}$$

which are homomorphisms.

Let us observe that given a cell $\varsigma \in \mathcal{D}(B)$, $\varsigma \neq \Upsilon$, the transformations $t$, $s$ and $u$ permute the preimages of $\varsigma$ under $P \circ Q$ among the spaces $\{B_{i,j,k}\}$, and these preimages are exactly all the translations of $\varsigma'$ by $t$, $s$ and $u$. Likewise, we may see that $u$ permutes the preimages of $\Upsilon$ under $P \circ Q$ among the spaces $\{B'_k\}$, and that these preimages are exactly all the $n$ translations of $\Upsilon'$ by $u$.

In general, given $\varsigma \in \mathcal{D}(B)$, the transformations $t$, $s$ and $u$ will act trivially or not over preimages of $\varsigma$ depending on whether or not $\varsigma$ belongs to the branching sets associated to each of these transformations, and on whether or not $\varsigma$ is $\Upsilon$. As before, $t$ acts trivially over all the preimages of $\Upsilon$ and all cells lying on $L_{x=0}$, and $s$ acts trivially over all the preimages of $\Upsilon$ and all cells lying on $L_{y=0}$. Additionally, $u$ acts trivially over all the preimages of cells lying on $\mathcal{H}_{1,1}$. Thus, we have the following:

**Lemma 3.16.** *The cells composing $\mathcal{D}(\mathcal{CF})$ are exactly the following, and satisfy the properties described. The term preimage refers to preimage under $P \circ Q$.*

- *The preimages of $\Upsilon$, namely $\Upsilon'$, $u(\Upsilon')$,…,$u^{n-1}(\Upsilon')$. Only $t$ and $s$ act trivially over these cells.*
- *The preimages of $P_1$, $c_1$, $p_1$, $\zeta_1$, $\hat{P}_1$ and $\hat{c}_1$, which are the cells of $\mathcal{D}(B)$ lying on $L_{x=0}$. If $\varsigma$ is one of these cells, its preimages are*
$$\left\{s^j u^k(\varsigma')\right\}_{0\le j\le b-1,\ 0\le k\le n-1}.$$
*Only $t$ acts trivially over these cells.*
- *The preimages of $P_2$, $c_2$, $p_2$, $\zeta_2$, $\hat{P}_2$ and $\hat{c}_2$, which are the cells of $\mathcal{D}(B)$ lying on $L_{y=0}$. If $\varsigma$ is one of these cells, its preimages are*
$$\left\{t^i u^k(\varsigma')\right\}_{0\le i\le a-1,\ 0\le k\le n-1}.$$
*Only $s$ acts trivially over these cells.*
- *The preimages of $Q_1$, $Q_2$, $h_1$, $h_2$, $q_1$, $q_2$, $\eta_1$, $\eta_2$, $\hat{R}$ and $\hat{k}$, which are the cells of $\mathcal{D}(B)$ lying on $\mathcal{H}_{1,1}$. If $\varsigma$ is one of these cells, its preimages are*
$$\left\{t^i s^j(\varsigma')\right\}_{0\le i\le a-1,\ 0\le j\le b-1}.$$
*Only $u$ acts trivially over these cells.*
- *The preimages of all the remaining cells of $\mathcal{D}(B)$. If $\varsigma$ is one of these cells, its preimages are*
$$\left\{t^i s^j u^k(\varsigma')\right\}_{0\le i\le a-1,\ 0\le j\le b-1,\ 0\le k\le n-1}.$$
*Neither $t$, $s$ or $u$ act trivially over these cells.*

Now we want to calculate the boundaries of all the cells of $\mathcal{D}(\mathcal{CF})$. The following lemma is a consequence of the definition of $\mathcal{D}(\mathcal{CF})$, $t$, $s$ and $u$.

**Lemma 3.17.** *For every $\varsigma \in \mathcal{D}(\mathcal{CF})$, and every $i, j, u \in \mathbb{Z}$,*

$$\partial(t^i s^j u^k \varsigma) = t^i s^j u^k \partial(\varsigma)\,.$$

Lemma 3.16 implies that, once the boundaries of the cells of $\mathcal{B}$ have been calculated, this formula provides us the boundaries of all the cells of $\mathcal{D}(\mathcal{CF})$. The boundaries of the cells of $\mathcal{B}$ are given below.

Dimension 1:

*Upper*

$$\begin{array}{ll}
\partial(m) = usR - R & \\
\partial(l) = utR - R & \\
\partial(k) = tR - sR & \\
\partial(h_1) = tQ_1 - sQ_1 & \partial(h_2) = sQ_2 - tQ_2 \\
\partial(c_1) = tP_1 - P_1 & \partial(c_2) = sP_2 - P_2 \\
\partial(a_1) = P_1 - Q_1 & \partial(a_2) = P_2 - Q_2 \\
\partial(b_1) = Q_1 - R & \partial(b_2) = Q_2 - R
\end{array}$$

*Middle*

$$\begin{array}{ll}
\partial(p_1) = P_1 - \hat{P}_1 & \partial(p_2) = P_2 - \hat{P}_2 \\
\partial(q_1) = Q_1 - \hat{R} & \partial(q_2) = Q_2 - \hat{R} \\
\partial(r) = R - \hat{R} &
\end{array}$$

*Lower*

$$\begin{array}{lll}
\partial(\hat{m}) = s\hat{R} - \hat{R} & \partial(\hat{c}_1) = t\hat{P}_1 - \hat{P}_1 & \partial(\hat{c}_2) = s\hat{P}_2 - \hat{P}_2 \\
\partial(\hat{l}) = t\hat{R} - \hat{R} & \partial(\hat{a}_1) = \hat{P}_1 - \hat{R} & \partial(\hat{a}_2) = \hat{P}_2 - \hat{R} \\
\partial(\hat{k}) = t\hat{R} - s\hat{R} & &
\end{array}$$

Dimension 2:

*Upper*

$$\begin{array}{ll}
\partial(\sigma) = sl - tm - k & \\
\partial(\pi) = m - uk - l & \\
\partial(\theta_1) = m + sa_1 + usb_1 - a_1 - b_1 & \partial(\theta_2) = l + ta_2 + utb_2 - a_2 - b_2 \\
\partial(\omega_1) = c_1 - l + a_1 + b_1 - ta_1 - tub_1 & \partial(\omega_2) = c_2 - m + a_2 + b_2 - sa_2 - sub_2 \\
\partial(\phi_1) = h_1 - k + sb_1 - tb_1 & \partial(\phi_2) = h_2 + k + tb_2 - sb_2
\end{array}$$

*Middle*

$$\begin{array}{ll}
\partial(\mu) = usr - r + \hat{m} - m & \\
\partial(\lambda) = utr - r + \hat{l} - l & \\
\partial(\kappa) = tr - sr + \hat{k} - k & \\
\partial(\eta_1) = tq_1 - sq_1 + \hat{k} - h_1 & \partial(\eta_2) = sq_2 - tq_2 - \hat{k} - h_2 \\
\partial(\zeta_1) = tp_1 - p_1 + \hat{c}_1 - c_1 & \partial(\zeta_2) = sp_2 - p_2 + \hat{c}_2 - c_2 \\
\partial(\alpha_1) = p_1 - q_1 + \hat{a}_1 - a_1 & \partial(\alpha_2) = p_2 - q_2 + \hat{a}_2 - a_2 \\
\partial(\beta_1) = q_1 - r - b_1 & \partial(\beta_2) = q_2 - r - b_2
\end{array}$$

*Lower*

$$\begin{array}{ll}
\partial(\hat{\sigma}) = s\hat{l} - t\hat{m} - \hat{k} & \\
\partial(\hat{\pi}) = \hat{m} + u\hat{k} - \hat{l} & \\
\partial(\hat{\theta}_1) = \hat{m} + s\hat{a}_1 - \hat{a}_1 & \partial(\hat{\theta}_2) = \hat{l} + t\hat{a}_2 - \hat{a}_2 \\
\partial(\hat{\omega}_1) = \hat{c}_1 - \hat{l} + \hat{a}_1 - t\hat{a}_1 & \partial(\hat{\omega}_2) = \hat{c}_2 - \hat{m} + \hat{a}_2 - s\hat{a}_2
\end{array}$$

Dimension 3:

*Upper*

$$\begin{aligned}
&\partial(\Psi_1) = \sigma + \pi + t\theta_1 - \theta_1 + s\omega_1 - \omega_1 + u\phi_1 - \phi_1 \\
&\partial(\Psi_2) = -\sigma - \pi + s\theta_2 - \theta_2 + t\omega_2 - \omega_2 + u\phi_2 - \phi_2
\end{aligned}$$

*Middle*

$$\begin{aligned}
&\partial(\Theta_1) = \mu + s\alpha_1 + us\beta_1 - \alpha_1 - \beta_1 + \theta_1 - \hat{\theta}_1 \\
&\partial(\Theta_2) = \lambda + t\alpha_2 + ut\beta_2 - \alpha_2 - \beta_2 + \theta_2 - \hat{\theta}_2 \\
&\partial(\Omega_1) = \zeta_1 - \lambda + \alpha_1 + \beta_1 - t\alpha_1 - ut\beta_1 + \omega_1 - \hat{\omega}_1 \\
&\partial(\Omega_2) = \zeta_2 - \mu + \alpha_2 + \beta_2 - s\alpha_2 - us\beta_2 + \omega_2 - \hat{\omega}_2 \\
&\partial(\Phi_1) = \eta_1 - \kappa + s\beta_1 - t\beta_1 + \phi_1 \\
&\partial(\Phi_2) = \eta_2 + \kappa + t\beta_2 - s\beta_2 + \phi_2 \\
&\partial(\Sigma) = s\lambda - t\mu - \kappa + \sigma - \hat{\sigma} \\
&\partial(\Pi) = \mu + u\kappa - \lambda + \pi - \hat{\pi}
\end{aligned}$$

*Lower*

$$\begin{aligned}
&\partial(\hat{\Psi}_1) = \hat{\sigma} + \hat{\pi} + t\hat{\theta}_1 - \hat{\theta}_1 + s\hat{\omega}_1 - \hat{\omega}_1 \\
&\partial(\hat{\Psi}_2) = -\hat{\sigma} - \hat{\pi} + s\hat{\theta}_2 - \hat{\theta}_2 + t\hat{\omega}_2 - \hat{\omega}_2
\end{aligned}$$

Dimension 4:

$$\begin{aligned}
&\partial(\Xi_1) = \Sigma + \Pi + t\Theta_1 - \Theta_1 + s\Omega_1 - \Omega_1 + u\Phi_1 - \Phi_1 - \Psi_1 + \hat{\Psi}_1 \\
&\partial(\Xi_1) = -\Sigma - \Pi + s\Theta_2 - \Theta_2 + t\Omega_2 - \Omega_2 + u\Phi_2 - \Phi_2 - \Psi_2 + \hat{\Psi}_2 \\
&\partial(\Upsilon) = (1 + t + \cdots + t^{a-1})(1 + s + \cdots + s^{b-1})(\hat{\Psi}_1 - \hat{\Psi}_2)
\end{aligned}$$

## Chapter 4

# The Complex Homology of the Milnor Fiber

Let $f(x, y, z) = z^n - x^a y^b$ and $\mathcal{CF}$ be as in the previous chapter. Our purpose now is to calculate, for arbitrary $a$, $b$ and $n$, the complex homology of the compact Milnor fiber $\mathcal{CF}$. The main theorem of the chapter is the following.

**Theorem 4.1.** The complex homology of $\mathcal{CF}$ is the following:

$$\begin{array}{rcl} H_0(\mathcal{CF};\mathbb{C}) & = & \mathbb{C}, \\ H_1(\mathcal{CF};\mathbb{C}) & = & \mathbb{C}^{(d-1)(n-1)}, \\ H_2(\mathcal{CF};\mathbb{C}) & = & \mathbb{C}^{(d-1)(n-1)+n-1}, \\ H_3(\mathcal{CF};\mathbb{C}) & = & 0, \\ H_4(\mathcal{CF};\mathbb{C}) & = & 0. \end{array}$$

Where $d := \gcd(a, b)$.

The next sections are devoted to the proof of this theorem. Along them, $i$, $j$, and $k$ are always thought as integers modulo $a$, $b$ and $n$ respectively.

## 4.1 Preliminaries

Let us define

$$R := \mathbb{C}[\mathbf{t},\mathbf{s},\mathbf{u}] \big/ \mathbf{t}^a - 1,\ \mathbf{s}^b - 1,\ \mathbf{u}^n - 1.$$

We write the formal variables $\mathbf{t}$, $\mathbf{s}$ and $\mathbf{u}$ in boldface to distinguish them from the deck transformations $t$, $s$, and $u$. Let us notice that besides being a ring, $R$ is also a $\mathbb{C}$-vector space. As a vector space, $R$ has dimension $abn$, and the natural basis

$$\left\{\mathbf{t}^i\mathbf{s}^j\mathbf{u}^k\right\}_{i\in[[0,a-1]],\ j\in[[0,b-1]],\ k\in[[0,n-1]]},$$

however, we will define another basis that will be more convenient for us. Let us fix the following notation:

$$\zeta_l := e^{2\pi i\frac{l}{a}},\quad \xi_j := e^{2\pi i\frac{j}{b}},\quad \mu_k := e^{2\pi i\frac{k}{n}}.$$

And for $0 \leq i \leq a-1$, $0 \leq j \leq b-1$ and $0 \leq k \leq n-1$, let us define

$$\begin{aligned} p_i(\mathbf{t}) &:= \frac{1+(\bar{\zeta}_i\mathbf{t})^1+\cdots+(\bar{\zeta}_i\mathbf{t})^{a-1}}{\zeta_i} = \frac{\mathbf{t}^a-1}{\mathbf{t}-\zeta_i},\\ p_j(\mathbf{s}) &:= \frac{1+(\bar{\xi}_j\mathbf{s})^1+\cdots+(\bar{\xi}_j\mathbf{s})^{b-1}}{\xi_j} = \frac{\mathbf{s}^b-1}{\mathbf{s}-\xi_j},\\ p_k(\mathbf{u}) &:= \frac{1+(\bar{\mu}_k\mathbf{u})^1+\cdots+(\bar{\mu}_k\mathbf{u})^{n-1}}{\mu_k} = \frac{\mathbf{u}^n-1}{\mathbf{u}-\mu_k}. \end{aligned}$$

Then, the set

$$\{p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})\}_{i\in[[0,a-1]],\ j\in[[0,b-1]],\ k\in[[0,n-1]]}$$

is a basis of $R$, as we are about to prove.

Let us define

$$\begin{aligned} \times_{\mathbf{t}} : R &\longrightarrow R, & \times_{\mathbf{s}} : R &\longrightarrow R, & \times_{\mathbf{u}} : R &\longrightarrow R\\ q &\longmapsto \mathbf{t}q & q &\longmapsto \mathbf{s}q & q &\longmapsto \mathbf{u}q \end{aligned}.$$

Let us observe that, for every $i$, $(\mathbf{t}-\zeta_i)p_i(\mathbf{t}) = 0$, which implies that $\zeta_i$ is an eigenvalue of $\times_{\mathbf{t}}$, and $p_i(\mathbf{t})$ an eigenvector associated with $\zeta_i$. Similarly, for every $j$ and $k$, $p_j(\mathbf{s})$ is an eigenvector of $\times_{\mathbf{s}}$ associated with the eigenvalue $\xi_j$, and $p_i(\mathbf{u})$ an eigenvector of $\times_{\mathbf{u}}$ associated with the eigenvalue $\mu_k$. Therefore, every element $p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})$ is simultaneously an eigenvector of $\times_{\mathbf{t}}$, $\times_{\mathbf{s}}$ and $\times_{\mathbf{u}}$, associated with eigenvalues $\zeta_i$, $\xi_j$ and $\mu_k$ respectively.

**Lemma 4.2.** *The set $\{p_i(\boldsymbol{t})p_j(\boldsymbol{s})p_k(\boldsymbol{u})\}$ is a basis for $R$ as a $\mathbb{C}$-vector space.*

*Proof.* Given an eigenvalue $\zeta_i$ of $\times_{\mathbf{t}}$, we denote the eigenspace associated with $\zeta_i$ by $E_t(\zeta_i)$, and similarly for eigenvalues $\xi_j$ and $\mu_k$. Now, let us fix $k = 0$. For every $j$, the set of vectors

$$X_j := \{p_0(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u}), \ldots, p_{a-1}(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u})\}$$

is linearly independent, since it is formed by eigenvectors of $\times_{\mathbf{t}}$ associated with different eigenvalues, namely 1, $\zeta_1,\ldots,\zeta_{a-1}$.

On the other hand, given $j$ and $j'$, the spaces $E_s(\xi_j)$ and $E_s(\xi_{j'})$ intersect only at the origin. Since, for every $j$, $X_j \subset E_s(\xi_j)$, it follows that $X_0 \cup \cdots \cup X_{b-1}$ is a L.I. set. Yet $X_0 \cup \cdots \cup X_{b-1}$ is contained in $E_u(\mu_0)$, and by repeating this argument we may show that

$\{p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})\}$ is a L.I. set. Since this is a set of *abn* vectors, we have proved that it is a basis. □

The following corollary is clear from the proof.

**Corollary 4.3.** *For every $i$, $j$ and $k$, the sets*

$$\begin{aligned}&\{p_i(\mathbf{t})p_0(\mathbf{s})p_0(\mathbf{u}),\ldots,p_i(\mathbf{t})p_{b-1}(\mathbf{s})p_{n-1}(\mathbf{u})\}\,,\\&\{p_0(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u}),\ldots,p_{a-1}(\mathbf{t})p_j(\mathbf{s})p_{n-1}(\mathbf{u})\}\,,\text{ and}\\&\{p_0(\mathbf{t})p_0(\mathbf{s})p_k(\mathbf{u}),\ldots,p_{a-1}(\mathbf{t})p_{b-1}(\mathbf{s})p_k(\mathbf{u})\}\end{aligned}$$

*are bases for $E_t(\zeta_i)$, $E_s(\xi_j)$ and $E_u(\mu_k)$ respectively. A basis for any intersection of these eigenspaces is obtained by intersecting these bases in the same way. In particular, for fixed $i$, $j$ and $k$ the set*

$$\{p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})\}$$

*is a basis for $E_t(\zeta_i)\cap E_s(\xi_j)\cap E_u(\mu_k)$.*

Now, let us define

$$\begin{aligned}R_{t,s,u} &:= R,\\ R_{t,s} &:= R\big/\mathbf{u}-1,\\ R_{t,u} &:= R\big/\mathbf{s}-1,\\ &\ \vdots\\ R_u &:= R\big/\mathbf{t}-1,\ \mathbf{s}-1,\\ R_\emptyset &:= R\big/\mathbf{t}-1,\ \mathbf{s}-1,\ \mathbf{u}-1.\end{aligned}$$

We can also find bases of eigenvectors for these spaces as we did for $R$.

**Lemma 4.4.** *The following sets are bases for the respective $\mathbb{C}$-vector spaces:*

$$\begin{array}{llll}\{p_i(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u})\}_{i\in[[0,a-1]],\ j\in[[0,b-1]]} & (\textit{or }\ \{p_i(\mathbf{t})p_j(\mathbf{s})\}\,) & \textit{for} & R_{t,s},\\ \{p_i(\mathbf{t})p_0(\mathbf{s})p_k(\mathbf{u})\}_{i\in[[0,a-1]],\ k\in[[0,n-1]]} & (\textit{or }\ \{p_i(\mathbf{t})p_k(\mathbf{u})\}\,) & \textit{for} & R_{t,u},\\ \vdots & \vdots & \vdots & \vdots\\ \{p_0(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u})\}_{j\in[[0,b-1]]} & (\textit{or }\ \{p_j(\mathbf{s})\}\,) & \textit{for} & R_s,\\ \{p_0(\mathbf{t})p_0(\mathbf{s})p_k(\mathbf{u})\}_{k\in[[0,n-1]]} & (\textit{or }\ \{p_k(\mathbf{u})\}\,) & \textit{for} & R_u,\\ \{p_0(\mathbf{t})p_0(\mathbf{s})p_0(\mathbf{u})\} & (\textit{or }\ \{1\}\,) & \textit{for} & R_\emptyset.\end{array}$$

*Proof.* Let us consider the case of $R_{t,s}$. In this space, the class $[p_k(\mathbf{u})]$ of $p_k(\mathbf{u})$ satisfies the following:

$$\begin{array}{lll}\text{if } k=0 & \text{then} & \mu_k=1 \text{ and } [p_k(\mathbf{u})]=[n],\\ \text{if } k\neq 0 & \text{then} & [p_k(\mathbf{u})]=[0].\end{array}$$

This can be easily confirmed: The division algorithm tells us that $p_k(\mathbf{u}) = (\mathbf{u}-1)q(\mathbf{u}) + p_k(1)$. If $k = 0$, then $p_k(\mathbf{u}) = \frac{\mathbf{u}^{n}-1}{\mathbf{u}-1} = 1+\mathbf{u}+\cdots+\mathbf{u}^{n-1}$. Therefore, $p_k(1) = n$ and $[p_k(\mathbf{u})] = [p_k(1)] = [n]$. On the other hand, let us recall that

$$p_k(\mathbf{u}) = (\mathbf{u}-1)(\mathbf{u}-\mu_1)\cdots\underline{(\mathbf{u}-\mu_k)}\cdots(\mathbf{u}-\mu_{n-1}),$$

where notation $\underline{\cdot}$ means omission. If $k \neq 0$, then $p_k(\mathbf{u})$ has a factor of the form $(\mathbf{u}-1)$, which implies that $[p_k(\mathbf{u})] = [0]$.

Now, a generating set for $R_{t,s}$ can be obtained by taking the classes in $R_{t,s}$ of the elements of a basis of $R$. Applying this procedure to $\{p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})\}$ we obtain the generating set $\{p_i(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u})\} = \{p_i(\mathbf{t})p_j(\mathbf{s})n\}$ for $R_{t,s}$. By arguments similar to those of the previous lemma we can see that this set is in fact a basis. Besides, the factor $n$ can be ignored because it is a constant. The remaining cases can be handled analogously. □

On the other hand, let

$$\mathbb{C}_{i,j,k} := \mathbb{C}[\mathbf{t},\mathbf{s},\mathbf{u}]\big/\mathbf{t}-\zeta_i,\ \mathbf{s}-\xi_j,\ \mathbf{u}-\mu_k.$$

We will prove now some facts concerning these spaces and their relation with $R$.

**Lemma 4.5.** *For every $i$, $j$ and $k$, $\mathbb{C}_{i,j,k}$ is isomorphic to $\mathbb{C}$.*

*Proof.* By the division algorithm, for $c \in \mathbb{C}$, $p = (x-c)q + p(c)$, for some $q \in \mathbb{C}[x]$. The class $[p]$ of $p$ in $\mathbb{C}[x]\big/(x-c)$ is defined by

$$[p] = \{(x-c)q + p(c) \mid q \in \mathbb{C}[x]\}.$$

Thus, the correspondence $[p] \leftrightarrow p(c)$ is an isomorphism between $\mathbb{C}[x]\big/x-c$ and $\mathbb{C}$. The statement of the lemma follows from here inductively. The isomorphism between $\mathbb{C}_{i,j,k}$ and $\mathbb{C}$ is given by $[p] \leftrightarrow p(\zeta_i, \xi_j, \mu_k)$. □

**Observation 4.6.** For every $i$, $j$ and $k$, $\mathbb{C}_{i,j,k}$ is isomorphic to $p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})R$.

*Proof.* Let $q \in R$. By applying the division algorithm to $q$, successively dividing by $\mathbf{t}-\zeta_i$, $\mathbf{s}-\xi_j$ and $\mathbf{u}-\mu_k$, we find that the equality

$$p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})q = p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})q(\zeta_i, \xi_j, \mu_k)$$

holds in $R$. From here, it follows that the correspondence $p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})q \leftrightarrow q(\zeta_i, \xi_j, \mu_k)$ is an isomorphism between $p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})R$ and $\mathbb{C}$. Since $\mathbb{C}_{i,j,k}$ is isomorphic to $\mathbb{C}$, we have obtained the result. The isomorphism between $\mathbb{C}_{i,j,k}$ and $p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})R$ is given by the correspondence $[q] \leftrightarrow p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})q$. □

Given any $q \in \mathbb{C}[\mathbf{t},\mathbf{s},\mathbf{u}]$, we denote the class of $q$ in $\mathbb{C}_{i,j,k}$ by $[q]_{i,j,k}$. We shall notice that, given a fixed $\mathbb{C}_{i,j,k}$, the class of some $p_{i'}(\mathbf{t})$ in $\mathbb{C}_{i,j,k}$ satisfies that

$$[p_{i'}(\mathbf{t})]_{i,j,k} = [0]_{i,j,k} \text{ if and only if } i' \neq i.$$

This can be seen by an already familiar argument. Let us recall that

$$p_{i'}(\mathbf{t}) = (\mathbf{t}-1)(\mathbf{t}-\zeta_1)\cdots\underline{(\mathbf{t}-\zeta_{i'})}\cdots(\mathbf{t}-\zeta_{a-1}).$$

If $i' = i$, then $p_{i'}(\mathbf{t})$ has no factor of the form $(\mathbf{t}-\zeta_i)$, and since $p_{i'}(\mathbf{t})$ has no factors of the form $(\mathbf{s}-\xi_j)$ and $(\mathbf{u}-\mu_k)$ either, then $[p_{i'}(\mathbf{t})]_{i,j,k} \neq [0]_{i,j,k}$ in $\mathbb{C}_{i,j,k}$ by definition. On the other hand, if $i' \neq i$, then $p_{i'}(\mathbf{t})$ has a factor of the form $(\mathbf{t}-\zeta_i)$, which implies that $[p_{i'}(\mathbf{t})]_{i,j,k} = [0]_{i,j,k}$.

Similarly,

$$\begin{array}{rcl} [p_{j'}(\mathbf{s})]_{i,j,k} & = & [0]_{i,j,k} \text{ if and only if } j' \neq j \text{ and} \\ [p_{k'}(\mathbf{u})]_{i,j,k} & = & [0]_{i,j,k} \text{ if and only if } k' \neq k. \end{array}$$

**Lemma 4.7.** *The space $R$ is isomorphic to the direct sum expressed below, both as a ring and as a $\mathbb{C}$-vector space.*

$$R \approx \bigoplus_{\substack{i\in[[0,a-1]] \\ j\in[[0,b-1]] \\ k\in[[0,n-1]]}} \mathbb{C}_{i,j,k}$$

*Proof.* We define an isomorphism $\varphi : R \longrightarrow \oplus\,\mathbb{C}_{i,j,k}$ by defining each of its components. The component $\varphi_{i,j,k}$ of $\varphi$ on a given $\mathbb{C}_{i,j,k}$ is given by

$$\varphi_{i,j,k}(q) := [q]_{i,j,k} = [q(\zeta_i, \xi_j, \mu_k)]_{i,j,k}.$$

Let $p_{i'}(\mathbf{t})p_{j'}(\mathbf{s})p_{k'}(\mathbf{u})$ be a basic vector of $R$. Then,

$$\begin{array}{rcl} \varphi_{i,j,k}(p_{i'}(\mathbf{t})p_{j'}(\mathbf{s})p_{k'}(\mathbf{u})) & = & [p_{i'}(\mathbf{t})p_{j'}(\mathbf{s})p_{k'}(\mathbf{u})]_{i,j,k} \\ & = & [p_{i'}(\mathbf{t})]_{i,j,k}[p_{j'}(\mathbf{s})]_{i,j,k}[p_{k'}(\mathbf{u})]_{i,j,k}, \end{array}$$

and since $[p_{i'}(\mathbf{t})]_{i,j,k} = [0]_{i,j,k}$ if and only if $i' \neq i$, and similarly for $p_{j'}(\mathbf{s})$ and $p_{k'}(\mathbf{u})$, it holds that

$$\begin{array}{ll} \varphi_{i,j,k}(p_{i'}(\mathbf{t})p_{j'}(\mathbf{s})p_{k'}(\mathbf{u})) = [0]_{i,j,k} & \text{if} \quad (i',j',k') \neq (i,j,k) \text{ and} \\ \varphi_{i,j,k}(p_{i'}(\mathbf{t})p_{j'}(\mathbf{s})p_{k'}(\mathbf{u})) = [c]_{i,j,k} & \text{if} \quad (i',j',k') = (i,j,k), \end{array}$$

where $c$ is a constant. Therefore,

$$\varphi(p_{i'}(\mathbf{t})p_{j'}(\mathbf{s})p_{k'}(\mathbf{u})) = (0,\ldots,0,\underbrace{[c]_{i',j',k'}}_{\text{Position } i',j',k'},0,\ldots,0).$$

This implies that, for every $i$, $j$ and $k$, $\varphi$ sends $\langle p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})\rangle$ isomorphically into $0 \oplus \cdots \oplus 0 \oplus \mathbb{C}_{i,j,k} \oplus 0 \oplus \cdots \oplus 0$. From here, it follows that $\varphi$ is an isomorphism. $\square$

## 4.2 The Chain Spaces and Their Bases

For every $j \in \{0, \ldots, 4\}$, let $X^{(j)}$ be the set of cells of $\mathcal{D}(\mathcal{CF})$ of dimension $j$, $B^{(j)} := X^{(j)} \cap \mathcal{B}$, and $M^{(j)}$ the $\mathbb{C}$-vector space generated by $X^{(j)}$, this is, the space of formal complex linear combinations of elements of $X^{(j)}$.

Then, $M^{(j)}$ has a $R$-module structure given by the following operation: for $q = a_1\mathbf{t}^{k_1}\mathbf{s}^{k_2}\mathbf{u}^{k_3} + \cdots + a_0 \in R$ and $\varsigma \in X^{(j)}$,

$$q\varsigma := a_1 t^{k_1} s^{k_2} u^{k_3}(\varsigma) + \cdots a_0(\varsigma),$$

where $t$, $s$ and $u$ are not variables, but the deck transformations already defined. We denote $M^{(j)}$ when considering it with this $R$-module structure as $RM^{(j)}$. Lemma 3.16 implies that $B^{(j)}$ is a generating set for $RM^{(j)}$.

Let $j$ remain fixed for the rest of the section, and let us denote $X^{(j)}$, $B^{(j)}$ and $M^{(j)}$ simply by $X$, $B$ and $M$. Then, for every $* \subset \{t, s, u\}$, let us define

$$\begin{aligned} X_* :=& \quad \{\varsigma \in X \mid g \in \{t, s, u\} \text{ acts trivially over } \varsigma \textit{ iff } g \notin *\}\,, \\ B_* :=& \quad \{\varsigma \in B \mid g \in \{t, s, u\} \text{ acts trivially over } \varsigma \textit{ iff } g \notin *\}\,. \end{aligned}$$

In other words, $X_*$ (res. $B_*$) is the subset of $X$ (res. $B$) formed by the cells that are translated by the transformations of $*$, and remain fixed by the rest of the transformations. Then,

$$\begin{aligned} X &= X_{t,s,u} \cup X_{t,s} \cup X_{t,u} \cup X_{s,u} \cup X_t \cup X_s \cup X_u \cup X_\emptyset \text{ and} \\ B &= B_{t,s,u} \cup B_{t,s} \cup B_{t,u} \cup B_{s,u} \cup B_t \cup B_s \cup B_u \cup B_\emptyset. \end{aligned}$$

For $* \subset \{t, s, u\}$, let $M_*$ be the $\mathbb{C}$-vector space generated by $X_*$. Then, $M_*$ is a subspace of $M$, and also an $R$-submodule of $RM$. We denote $M_*$ when considering it with this submodule structure as $RM_*$. Then,

$$\begin{aligned} M &= M_{t,s,u} \oplus M_{t,s} \oplus M_{t,u} \oplus M_{s,u} \oplus M_t \oplus M_s \oplus M_u \oplus M_\emptyset \text{ and} \\ RM &= RM_{t,s,u} \oplus RM_{t,s} \oplus RM_{t,u} \oplus RM_{s,u} \oplus RM_t \oplus RM_s \oplus RM_u \oplus RM_\emptyset. \end{aligned}$$

This decomposition has the defect that the submodules $RM_*$ are not $R$-free (except for the first one). However, this can be easily solved. For $* \subset \{t, s, u\}$, let $R_*M_*$ denote the $R_*$-module freely generated by $B_*$.

Let us consider $RM_{t,s}$ for a moment. Since $u$ acts trivially over every cell of $M_{t,s}$, it holds that $q(\mathbf{t},\mathbf{s},\mathbf{u})\varsigma = q(\mathbf{t},\mathbf{s}, 1)\varsigma$, for every $q(\mathbf{t},\mathbf{s},\mathbf{u}) \in R$ and every $\varsigma \in M_{t,s}$. This implies that $RM_{t,s} = R_{t,s}M_{t,s}$. A similar situation stands for every $RM_*$, and therefore

$$RM = RM_{t,s,u} \oplus R_{t,s}M_{t,s} \oplus R_{t,u}M_{t,u} \oplus R_{s,u}M_{s,u} \oplus R_tM_t \oplus R_sM_s \oplus R_uM_u \oplus R_\emptyset M_\emptyset.$$

Now we need to find a basis for $M$ and each of the $M_*$. Although $X$ and every $X_*$ are such bases by definition, we will define another basis that will be more convenient for us.

Given that $M_*$ is a free $R_*$-module with basis $B_*$, we know that $M_* = \oplus_{b \in B_*} R_* b$. Hence, if $G$ is a basis for $R_*$ as a $\mathbb{C}$-vector space, then $GB_*$ is a basis for $M_*$ as a $\mathbb{C}$-vector space. By Lemmas 4.2 and 4.4, we have the following.

**Lemma 4.8.** *The following sets are bases for the respective vector spaces.*

$$\begin{array}{lll}
\{p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})x_{t,s,u}\}_{x_{t,s,u}\in B_{t,s,u}} & \textit{for} & M_{t,s,u}, \\
\{p_i(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u})x_{t,s}\}_{x_{t,s}\in B_{t,s}} & \textit{for} & M_{t,s}, \\
\{p_i(\mathbf{t})p_0(\mathbf{s})p_k(\mathbf{u})x_{t,u}\}_{x_{t,u}\in B_{t,u}} & \textit{for} & M_{t,u}, \\
\vdots & \vdots & \vdots \\
\{p_0(\mathbf{t})p_j(\mathbf{s})p_0(\mathbf{u})x_{s}\}_{x_{s}\in B_{s}} & \textit{for} & M_{s}, \\
\{p_0(\mathbf{t})p_0(\mathbf{s})p_k(\mathbf{u})x_{u}\}_{x_{u}\in B_{u}} & \textit{for} & M_{u}, \\
\{p_0(\mathbf{t})p_0(\mathbf{s})p_0(\mathbf{u})x_{\emptyset}\}_{x_{\emptyset}\in B_{\emptyset}} & \textit{for} & M_{\emptyset}.
\end{array}$$

*The union of these bases, that we will denote by $V$, is a basis for $M$.*

To have an adequate notation for these vectors, we define a bijection $h$ by the following rule:

$$\begin{array}{lll}
\{t,s,u\} & \longrightarrow & \{(i,j,k)\} \\
\{t,s\} & \longrightarrow & \{(i,j,0)\} \\
\{t,u\} & \longrightarrow & \{(i,0,k)\} \\
\vdots & \vdots & \vdots \\
\{s\} & \longrightarrow & \{(0,j,0)\} \\
\{u\} & \longrightarrow & \{(0,0,k)\} \\
\emptyset & \longrightarrow & \{(0,0,0)\}.
\end{array}$$

We also denote the power set of $\{t,s,u\}$ by $P(\{t,s,u\})$, and the polynomial $p_i(\mathbf{t})p_j(\mathbf{s})p_k(\mathbf{u})$ by $v_{i,j,k}$. Then, with this notation we can restate the previous lemma by saying that the following sets are bases for the respective $\mathbb{C}$-vector spaces:

$$\{v_{i,j,k}x_*\}_{(i,j,k)\in h(*)} = \bigcup_{(i,j,k)\in h(*)} v_{i,j,k}B_* \qquad \text{for} \quad M_*.$$

$$V := \{v_{i,j,k}x_* \mid * \in P(\{t,s,u\}),\, (i,j,k)\in h(*)\} \quad \text{for} \quad M.$$

For example, the basis for $M_{t,s}$ is $\{v_{i,j,0}x_{t,s}\}$, or equivalently $\bigcup_{i,j} v_{i,j,0}B_{t,s}$.

On the other hand, for simplicity, we keep calling $\times_{\mathbf{t}}$, $\times_{\mathbf{s}}$ and $\times_{\mathbf{u}}$ the transformations from $RM$ to $RM$ that multiply a given vector by $\mathbf{t}$, by $\mathbf{s}$, and by $\mathbf{u}$. These are in fact linear transformations from $M$ to $M$. We also keep denoting their eigenspaces by $E_t(\cdot)$, $E_s(\cdot)$ and $E_u(\cdot)$. We may use these eigenspaces to construct another decomposition of $M$. For each $i$, $j$ and $k$, let us define

$$M_{i,j,k} := E_t(\zeta_i) \cap E_s(\xi_j) \cap E_u(\mu_k).$$

We will find a basis for each of these subspaces and later prove that they decompose $M$ as a direct sum.

Let us recall that every element $v_{i,j,k} \in R$ is an eigenvector of $\times_{\mathbf{t}} : R \to R$, $\times_{\mathbf{s}} : R \to R$ and $\times_{\mathbf{u}} : R \to R$, associated with eigenvalues $\zeta_i$, $\xi_j$ and $\mu_k$ respectively. This situation is reflected both in $RM$ and $M$. For the same reason as in the case of $R$, every basic vector $v_{i,j,k}x_* \in V$ is an eigenvector of $\times_{\mathbf{t}} : M \to M$, $\times_{\mathbf{s}} : M \to M$ and $\times_{\mathbf{u}} : M \to M$, associated with eigenvalues $\zeta_i$, $\xi_j$ and $\mu_k$ respectively.

Moreover, from Corollary 4.3 it derives the following:

**Lemma 4.9.** *For fixed $i_0$, $j_0$ and $k_0$, the sets*

$$\begin{array}{l} \{v_{i_0,j,k}x_* \mid * \in P(\{t,s,u\}),\ (i_0,j,k) \in h(*)\}\,, \\ \{v_{i,j_0,k}x_* \mid * \in P(\{t,s,u\}),\ (i,j_0,k) \in h(*)\}\ \ \textit{and} \\ \{v_{i,j,k_0}x_* \mid * \in P(\{t,s,u\}),\ (i,j,k_0) \in h(*)\} \end{array}$$

*are bases for $E_t(\zeta_{i_0})$, $E_s(\xi_{j_0})$ and $E_u(\mu_{k_0})$ respectively. A basis for any intersection of these eigenspaces is obtained by intersecting these bases in the same way, and in particular, the set*

$$\{v_{i_0,j_0,k_0}x_* \mid * \in P(\{t,s,u\}),\ (i_0,j_0,k_0) \in h(*)\}$$

*is a basis for $M_{i_0,j_0,k_0}$.*

This basis for $M_{i_0,j_0,k_0}$ will be important for us later and will be denoted by $F_{i_0,j_0,k_0}$. Let us notice that if we define a set $B_{i,j,k} \subset B$ by

$$B_{i,j,k} := \bigcup_{\{*\mid(i,j,k)\in h(*)\}} B_*,$$

Then

$$F_{i,j,k} = v_{i,j,k}B_{i,j,k}.$$

Let $|\cdot|$ denote the cardinal of a set. The following will be useful observations.

**Remark 4.10.** $\dim(M_{i,j,k}) = |B_{i,j,k}|$.

**Remark 4.11.** For $i, j, k \neq 0$,

$$\begin{array}{rcl} B_{i,j,k} &=& B_{t,s,u}, \\ B_{i,j,0} &=& B_{t,s,u} \cup B_{t,s}, \\ B_{i,0,k} &=& B_{t,s,u} \cup B_{t,u}, \\ B_{0,j,k} &=& B_{t,s,u} \cup B_{s,u}, \\ B_{i,0,0} &=& B_{t,s,u} \cup B_{t,s} \cup B_{t,u} \cup B_t, \\ B_{0,j,0} &=& B_{t,s,u} \cup B_{t,s} \cup B_{s,u} \cup B_s, \\ B_{0,0,k} &=& B_{t,s,u} \cup B_{t,u} \cup B_{s,u} \cup B_u, \\ B_{0,0,0} &=& B. \end{array}$$

We will see now that the subspaces $M_{i,j,k}$ decompose $M$. For each $i$, $j$ and $k$ let us define

$$M_{*;i,j,k} := M_* \cap M_{i,,j,k}.$$

Then we have the following.

**Lemma 4.12.** *The following equalities hold:*

$$\begin{aligned} M &= \bigoplus_{i,j,k} M_{i,j,k}, \\ M_* &= \bigoplus_{i,j,k} M_{*;i,j,k}, \\ M_{i,j,k} &= \bigoplus_{*\in P(\{t,s,u\})} M_{*;i,j,k} \end{aligned}$$

To see the first equality it suffices to observe that

$$\begin{aligned} M &= \langle V \rangle \\ &= \bigoplus_{i,j,k} \langle \{ v_{i,j,k} x_* \mid * \in P(\{t,s,u\}),\, (i,j,k) \in h(*) \} \rangle \\ &= \bigoplus_{i,j,k} M_{i,j,k}. \end{aligned}$$

The rest of equalities follow from the first one. We finish this section by finding bases for $M_{*;i,j,k}$.

**Lemma 4.13.** *Given fixed $i, j, k$, the following equivalences hold:*

$$\begin{array}{lcc} M_{t,s,u;i,j,k} \neq 0 & \textit{iff} & 0 = 0 \ \textit{(i.e. always)}, \\ M_{t,s;i,j,k} \neq 0 & \textit{iff} & k = 0, \\ M_{t,u;i,j,k} \neq 0 & \textit{iff} & j = 0, \\ \vdots & \vdots & \vdots \\ M_{s;i,j,k} \neq 0 & \textit{iff} & i = 0 \ \textit{and} \ k = 0, \\ M_{u;i,j,k} \neq 0 & \textit{iff} & i = 0 \ \textit{and} \ j = 0, \\ M_{\emptyset;i,j,k} \neq 0 & \textit{iff} & i = 0,\ j = 0 \ \textit{and} \ k = 0. \end{array}$$

*The following sets are bases for the respective non-empty spaces:*

$$\begin{array}{lll} v_{i,j,k} B_{t,s,u} & \textit{for} & M_{t,s,u;i,j,k}, \\ v_{i,j,0} B_{t,s} & \textit{for} & M_{t,s;i,j,0}, \\ v_{i,0,k} B_{t,u} & \textit{for} & M_{t,u;i,0,k}, \\ \vdots & \vdots & \vdots \\ v_{0,j,0} B_{s} & \textit{for} & M_{s;0,j,0}, \\ v_{0,0,k} B_{u} & \textit{for} & M_{u;0,0,k}, \\ v_{0,0,0} B_{\emptyset} & \textit{for} & M_{\emptyset;0,0,0}. \end{array}$$

*Proof.* Let us prove in general that for fixed $*_0$, $i_0$, $j_0$ and $k_0$ the set $v_{i_0,j_0,k_0}B_{*_0}$ is a basis for $M_{*_0;i_0,j_0,k_0}$. A basis for $M_{*_0;i_0,j_0,k_0}$ is given by

$$\begin{aligned} & \{v_{i,j,k}x_{*_0}\}_{(i,j,k)\in h(*_0)} \cap \{v_{i_0,j_0,k_0}x_* \mid * \in P(\{t,s,u\}),\ (i_0,j_0,k_0)\in h(*)\} \\ = \quad & v_{i_0,j_0,k_0}B_{*_0}. \end{aligned}$$

From here, it follows the general equivalence

$$M_{*;i,j,k} \neq 0 \quad \text{iff} \quad (i,j,k) \in h(*),$$

from which the stated equivalences are particular cases. □

## 4.3 The Boundary Operator

Now we will work with different spaces $M^{(r)}$ at the same time. For each $M^{(r)}$ we will have then decompositions, bases, transformations and eigenspaces as defined in the previous sections. To distinguish between them, we will use the superindex $^{(r)}$, for example, $V^{(r)}$ and $V^{(r-1)}$ will denote bases for $M^{(r)}$ and $M^{(r-1)}$ respectively. Let us notice, however, that the transformations $\times^{(r)}\mathbf{t}$, $\times^{(r)}\mathbf{s}$ and $\times^{(r)}\mathbf{u}$ have always eigenvalues $\{\zeta_i\}$, $\{\xi_j\}$ and $\{\mu_k\}$ respectively, regardless of the value of $r$.

Let us consider the boundary operator $\partial^{(r)} : M^{(r)} \longrightarrow M^{(r-1)}$, which we will simply denote by $\partial$. Let $[\partial]$ denote the matrix of $\partial$ with respect to the bases $V^{(r)}$ and $V^{(r-1)}$. On the other hand, let $[\partial]_R$ be a matrix of $\partial$ on the generating sets $B^{(r)}$ and $B^{(r-1)}$, considering $RM^{(r)}$ and $RM^{(r-1)}$ as $R$-modules. Thus, $[\partial]$ is a matrix or complex numbers and $[\partial]_R$ a much smaller matrix of complex polynomials.

Let us recall that at the end of Section 3.7 we gave the boundaries of all the cells in $\mathcal{B}$. Due to our definition of product on $RM^{(r)}$, the symbols $t$, $s$ and $u$ written there can be thought either as deck transformations or as elements of $R$ multiplying the cells in the modules $RM^{(r)}$. According to the later approach, we see that the boundary of every cell of $B^{(r)}$ is given as a linear combination in $RM^{(r-1)}$ of elements of $B^{(r-1)}$. Therefore, to give the boundaries that we have given at the end of Section 3.7 is the same thing as giving the matrix $[\partial]_R$, though written in a different fashion. Hence, the matrix $[\partial]_R$ is explicitly known to us.

Now, an immediate consequence of Lemma 3.17 and our definition of product in $RM$ is that

$$\partial(\mathbf{t}^i\mathbf{s}^j\mathbf{u}^k\varsigma) = \mathbf{t}^i\mathbf{s}^j\mathbf{u}^k\partial(\varsigma).$$

As a consequence, if $v \in E_t^{(r)}(\zeta_i)$ for some $i$, then

$$\mathbf{t}\partial(v) = \partial(\mathbf{t}v) = \partial(\zeta_i v) = \zeta_i\partial(v),$$

which implies that $\partial(v) \in E_t^{(r-1)}(\zeta_i)$. By reasoning in a similar way for $\mathbf{s}$ and $\mathbf{u}$ we have the following.

**Lemma 4.14.** *For every $i$, $j$ and $k$,*

$$\begin{array}{rcl} \partial(E_t^{(r)}(\zeta i)) & \subset & E_t^{(r-1)}(\zeta i), \\ \partial(E_s^{(r)}(\xi_j)) & \subset & E_s^{(r-1)}(\xi_j), \\ \partial(E_u^{(r)}(\mu_k)) & \subset & E_u^{(r-1)}(\mu_k), \\ \partial(M_{i,j,k}^{(r)}) & \subset & M_{i,j,k}^{(r-1)}. \end{array}$$

It is easy to see also that $\partial(M_*^{(r)}) \subset M_*^{(r-1)}$ and $\partial(M_{*;i,j,k}^{(r)}) \subset M_{*;i,j,k}^{(r-1)}$, though we do not make use of this fact. Let us consider fixed $i$, $i'$, $j$, $j'$, $k$ and $k'$. For every $v \in V^{(r)}$, let $C_v$ denote the column of $[\partial]$ corresponding to the image of $v$; and let $C'_v$ be the vector containing the entries of $C_v$ associated with the elements of $F_{i',j',k'}^{(r-1)} \subset V^{(r-1)}$. Now we consider the set of vectors $\{C'_v \mid v \in F_{i,j,k}^{(r)}\}$. The matrix whose columns are the vectors on this set will be denoted by $[\partial_{i,j,k|i',j',k'}]$, or simply by $[\partial_{i,j,k}]$ if $i = i'$, $j = j'$ and $k = k'$.

Since for every $r$ the bases of the $M_{i,j,k}^{(r)}$, as given in Lemma 4.9, form a partition of $V^{(r)}$ (Lemma 4.12), by arranging the elements of $V^{(r)}$ and $V^{(r-1)}$ in an appropriate way we can decompose $[\partial]$ in disjoint submatrices of the form $[\partial_{i,j,k|i',j',k'}]$. The previous lemma implies that all these submatrices are zero except perhaps for those of the form $[\partial_{i,j,k}]$. By arranging the elements of $V^{(r)}$ and $V^{(r-1)}$ in an appropriate way, we may further locate these submatrices in the diagonal, for which we have the following lemma.

**Lemma 4.15.** *The matrix $[\partial]$ is "block diagonal" in the sense just described. A submatrix $[\partial_{i,j,k|i',j',k'}]$ is on the diagonal if and only if it is of the form $[\partial_{i,j,k}]$.*

Let us notice that, if we set $\partial_{i,j,k}$ to denote the restriction of $\partial$ to $M_{i,j,k}^{(r)}$ and $M_{i,j,k}^{(r-1)}$, then $[\partial_{i,j,k}]$ is exactly the matrix of $\partial_{i,j,k}$ on the bases $F_{i,j,k}$ and $F_{i,j,k}$. Similarly, $\partial_{i,j,k|i',j',k'}$ can be set to denote the composition of $\partial_{i,j,k}$ with the projection over $M_{i',j',k'}^{(r-1)}$.

Now we will show that $[\partial]_R$ can be used to find each $[\partial_{i,j,k}]$ explicitly. We will need to define yet another matrix, though by already familiar constructions. Let us consider fixed $i$, $j$ and $k$. For every $b \in B^{(r)}$, let $C_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u})$ denote the column of $[\partial]_R$ corresponding to the image of $b$; and let $C'_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u})$ be the vector containing the entries of $C_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u})$ associated with the elements of $B_{i,j,k}^{(r-1)}$. Now we consider the set of vectors $\{C'_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u}) \mid b \in B_{i,j,k}^{(r)}\}$. The matrix whose columns are the vectors on this set will be denoted by $[\partial_{i,j,k}]_R(\mathbf{t},\mathbf{s},\mathbf{u})$.

**Lemma 4.16.** *For every $i$, $j$ and $k$,*

$$[\partial_{i,j,k}] = [\partial_{i,j,k}]_R(\zeta_i, \xi_j, \mu_k).$$

*Proof.* We denote the vectors formed by the elements of $B^{(r-1)}$ and $B_{i,j,k}^{(r-1)}$ equally by

$B^{(r-1)}$ and $B^{(r-1)}_{i,j,k}$. Let $i$, $j$ and $k$ be fixed and $b \in B^{(r)}_{i,j,k}$, then

$$\begin{aligned}
\partial(v_{i,j,k}b) &= v_{i,j,k}\partial(b) \\
&= v_{i,j,k}C_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u}) \cdot B^{(r-1)} \\
&= v_{i,j,k}C'_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u}) \cdot B^{(r-1)}_{i,j,k} \quad \text{(by Lemma 4.14)} \\
&= C'_{R,b}(\mathbf{t},\mathbf{s},\mathbf{u}) \cdot v_{i,j,k}B^{(r-1)}_{i,j,k}.
\end{aligned}$$

But here, since the elements of $F_{i,j,k}$ belong to $E^{(r-1)}_t(\zeta_{i_0})$, $E^{(r-1)}_s(\xi_{j_0})$ and $E^{(r-1)}_k(\mu_{k_0})$, the last expression is equal to $C'_{R,b}(\zeta_i,\xi_j,\mu_k) \cdot v_{i,j,k}B^{(r-1)}_{i,j,k}$, and therefore

$$\partial(v_{i,j,k}b) = C'_{R,b}(\zeta_i,\xi_j,\mu_k) \cdot v_{i,j,k}B^{(r-1)}_{h(i,j,k)}.$$

This provides the result. $\square$

## 4.4 The Complex Homology

Our aim now will be to calculate the complex homology of $\mathcal{CF}$. Let

$$M^{(4)} \xrightarrow{\partial^{(4)}} M^{(3)} \xrightarrow{\partial^{(3)}} M^{(2)} \xrightarrow{\partial^{(2)}} M^{(1)} \xrightarrow{\partial^{(1)}} M^{(0)} \xrightarrow{\partial^{(0)}} 0$$

be the complex homology sequence of $\mathcal{CF}$. Since the chain spaces here are finite $\mathbb{C}$-vector spaces, all of them and their subspaces are direct sums of copies of $\mathbb{C}$. As a consequence, the spaces of boundaries, of cycles, and the homology spaces are all determined by their dimensions. All we need then is to calculate these dimensions.

The dimensions of the spaces of cycles and boundaries are given by the nullities and ranks of the matrices $[\partial^{(r)}]$, while the dimensions of the homology spaces are differences of these. Lemma 4.15 implies that the rank of $[\partial^{(r)}]$ is the sum of the ranks of the $[\partial^{(r)}_{i,j,k}]$, and similarly for the nullities. Therefore, all what is needed to calculate these dimensions is to calculate the rank and nullity of each matrix $[\partial^{(r)}_{i,j,k}]$, for $0 \le i \le a-1$, $0 \le j \le b-1$ and $0 \le k \le n-1$. Since these ranks will vary according to the values of $i$, $j$ and $k$, we will reason by cases.

Along the proofs, several calculations are done by using SageMath. The program used is included in appendix C. We will also need to know the cardinal of each set $B^{(r)}_*$. In order to find them, we present each $B^{(r)}$ and $B^{(r)}_*$ explicitly in the following table, which can be deduced from Lemma 3.16. The omitted sets are empty.

$$B^{(0)} = B^{(0)}_{t,s,u} \cup B^{(0)}_{t,s} \cup B^{(0)}_{t,u} \cup B^{(0)}_{s,u}$$

$$
\begin{aligned}
\text{with } B^{(0)}_{t,s,u} &= \{R\}\\
B^{(0)}_{t,s} &= \{Q_1, Q_2, \hat{R}\}\\
B^{(0)}_{t,u} &= \{P_1, \hat{P}_1\}\\
B^{(0)}_{s,u} &= \{P_2, \hat{P}_2\}
\end{aligned}
$$

$$B^{(1)} = B^{(1)}_{t,s,u} \cup B^{(1)}_{t,s} \cup B^{(1)}_{t,u} \cup B^{(1)}_{s,u}$$

$$
\begin{aligned}
\text{with } B^{(1)}_{t,s,u} &= \{m, l, k, a_1, a_2, b_1, b_2, \hat{m}, \hat{l}, \hat{a}_1, \hat{a}_2, r\}\\
B^{(1)}_{t,s} &= \{h_1, h_2, \hat{k}, q_1, q_2\}\\
B^{(1)}_{t,u} &= \{c_1, \hat{c}_1, p_1\}\\
B^{(1)}_{s,u} &= \{c_2, \hat{c}_2, p_2\}
\end{aligned}
$$

$$B^{(2)} = B^{(2)}_{t,s,u} \cup B^{(2)}_{t,s} \cup B^{(2)}_{t,u} \cup B^{(2)}_{s,u}$$

$$
\begin{aligned}
\text{with } B^{(2)}_{t,s,u} &= \{\sigma, \pi, \theta_1, \theta_2, \omega_1, \omega_2, \phi_1, \phi_2, \hat{\sigma}, \hat{\pi}, \hat{\theta}_1,\\
&\quad\ \hat{\theta}_2, \hat{\omega}_1, \hat{\omega}_2, \mu, \lambda, \kappa, \alpha_1, \alpha_2, \beta_1, \beta_2\}\\
B^{(2)}_{t,s} &= \{\eta_1, \eta_2, \}\\
B^{(2)}_{t,u} &= \{\zeta_1\}\\
B^{(2)}_{s,u} &= \{\zeta_2\}
\end{aligned}
$$

$$B^{(3)} = B^{(3)}_{t,s,u} = \{\Psi_1, \Psi_2, \hat{\Psi}_1, \hat{\Psi}_2, \Phi_1, \Phi_2, \Theta_1, \Theta_2, \Omega_1, \Omega_2, \Sigma, \Pi\}$$

$$B^{(4)} = B^{(4)}_{t,s,u} \cup B^{(4)}_{u}$$

$$
\begin{aligned}
\text{with } B^{(4)}_{t,s,u} &= \{\Xi_1, \Xi_2\}\\
B^{(4)}_{u} &= \{\Upsilon\}
\end{aligned}
$$

**Lemma 4.17.** *For $(i, j, k) = (0, 0, 0)$, the ranks and nullities of $\partial^{(0)}, \ldots, \partial^{(4)}$ are given by*

$$
\begin{array}{ll}
Null[\partial^{(0)}_{0,0,0}] = 8, & \\
Null[\partial^{(1)}_{0,0,0}] = 16, & Rk[\partial^{(1)}_{0,0,0}] = 7,\\
Null[\partial^{(2)}_{0,0,0}] = 9, & Rk[\partial^{(2)}_{0,0,0}] = 16,\\
Null[\partial^{(3)}_{0,0,0}] = 3, & Rk[\partial^{(3)}_{0,0,0}] = 9,\\
Null[\partial^{(4)}_{0,0,0}] = 0, & Rk[\partial^{(4)}_{0,0,0}] = 3.
\end{array}
$$

*Proof.* Let $r$ be fixed. Let us notice then that, since $B^{(r)}_{0,0,0} = B^{(r)}$, it holds that $[\partial^{(r)}_{0,0,0}]_R = [\partial^{(r)}]_R$, and by Lemma 4.16, $[\partial^{(r)}_{0,0,0}] = [\partial^{(r)}_{0,0,0}]_R(1,1,1) = [\partial^{(r)}]_R(1,1,1)$. Here, $[\partial^{(r)}]_R(1,1,1)$ is a matrix of complex numbers whose rank can be obtained by a straightforward calculation. We have calculated the rank of each $[\partial^{(r)}]_R(1,1,1)$ by using the program included in appendix C.

Now let us recall that by Remarks 4.10 and 4.11, the dimension of $M^{(r)}_{0,0,0}$ is equal to the cardinal of $B^{(r)}_{0,0,0} = B^{(r)}$. Therefore, for $r$ equal to 0, 1, 2, 3 and 4, $\dim(M^{(r)}_{0,0,0})$ is equal to 8, 23, 25, 12 and 3 respectively. The nullities can be calculated from here using the Rank Theorem. □

**Lemma 4.18.** *For $(i,0,0)$, with $i \neq 0$ the ranks and nullities of $\partial^{(0)}, \ldots, \partial^{(4)}$ are given by*

$$\begin{array}{ll}
Null[\partial^{(0)}_{i,0,0}] = 6, & \\
Null[\partial^{(1)}_{i,0,0}] = 14, & Rk[\partial^{(1)}_{i,0,0}] = 6, \\
Null[\partial^{(2)}_{i,0,0}] = 10, & Rk[\partial^{(2)}_{i,0,0}] = 14, \\
Null[\partial^{(3)}_{i,0,0}] = 2, & Rk[\partial^{(3)}_{i,0,0}] = 10, \\
Null[\partial^{(4)}_{i,0,0}] = 0, & Rk[\partial^{(4)}_{i,0,0}] = 2.
\end{array}$$

*Proof.* For any given matrix $A$, let $diag(A)$ and $S(A)$ denote the diagonal and Smith form of $A$ respectively. Let $r$ be fixed. We first use the program shown in Appendix C to find $[\partial^{(r)}_{i,0,0}]_R(\mathbf{t},\mathbf{s},\mathbf{u})$, by eliminating from $[\partial^{(r)}]_R$ the adequate rows and columns, and then to evaluate it in $\mathbf{s}=\mathbf{u}=1$. The entries of the resulting matrix $[\partial^{(r)}_{i,0,0}]_R(\mathbf{t},1,1)$ take values on the principal ideal domain $\mathbb{C}[\mathbf{t}]$, and therefore the Smith form for this matrix is defined. We then instruct the program to find the Smith form of $[\partial^{(r)}_{i,0,0}]_R(\mathbf{t},1,1)$. For each $r$, the diagonal of this Smith form, calculated by the program, is shown below.

$$\begin{array}{rcl}
diag\left(S\left([\partial^{(1)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\right) & = & (1,1,1,1,1,1), \\
diag\left(S\left([\partial^{(2)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\right) & = & (1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0), \\
diag\left(S\left([\partial^{(3)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\right) & = & (1,1,1,1,1,1,1,1,1,\mathbf{t}-1,0,0), \\
diag\left(S\left([\partial^{(4)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\right) & = & (1,1).
\end{array}$$

Now let us notice that, the only entry different from 1 and 0 appearing on any of the diagonals listed before is t−1, and for every $i \neq 0$, this entry satisfies that $(\mathbf{t}-1)\mid_{\mathbf{t}=\zeta_i} = \zeta_i - 1 \neq 0$. It is easily seen from here that the ranks of the matrices $[\partial^{(r)}_{i,0,0}]$ are given by the number of non-zero entries on these diagonals. Formally, we reason as follows. Since

$$S\left([\partial^{(r)}_{i,0,0}]_R(\zeta_i,1,1)\right) = S\left([\partial^{(r)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\mid_{\mathbf{t}=\zeta_i},$$

Then,

$$\begin{aligned} Rk\left([\partial^{(r)}_{i,0,0}]\right) &= Rk\left([\partial^{(r)}_{i,0,0}]_R(\zeta_i,1,1)\right), \\ &= Rk\left(S\left([\partial^{(r)}_{i,0,0}]_R(\zeta_i,1,1)\right)\right), \\ &= Rk\left(S\left([\partial^{(1)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\mid_{\mathbf{t}=\zeta_i}\right). \end{aligned}$$

This implies that, for each $r$, the rank of the matrix $[\partial^{(r)}_{i,0,0}]$ is the number of non-zero entries of $diag\left(S\left([\partial^{(r)}_{i,0,0}]_R(\mathbf{t},1,1)\right)\right)$.

By Remarks 4.10 and 4.11,

$$\dim(M^{(r)}_{i,0,0}) = \left|B^{(r)}_{i,0,0}\right| = \left|B^{(r)}_{t,s,u} \cup B^{(r)}_{t,s} \cup B^{(r)}_{t,u} \cup B^{(r)}_{t}\right|.$$

Therefore, for $r$ equal to 0, 1, 2, 3 and 4, $\dim(M^{(r)}_{i,0,0})$ is equal to 6, 20, 24, 12 and 2 respectively. The nullities can be calculated from here using the Rank Theorem. □

**Lemma 4.19.** *For* $(0,j,0)$*, with* $j \neq 0$ *the ranks and nullities of* $\partial^{(0)},\ldots,\partial^{(4)}$ *are given by*

$$\begin{aligned} &Null[\partial^{(0)}_{0,j,0}] = 6, && \\ &Null[\partial^{(1)}_{0,j,0}] = 14, && Rk[\partial^{(1)}_{0,j,0}] = 6, \\ &Null[\partial^{(2)}_{0,j,0}] = 10, && Rk[\partial^{(2)}_{0,j,0}] = 14, \\ &Null[\partial^{(3)}_{0,j,0}] = 2, && Rk[\partial^{(3)}_{0,j,0}] = 10, \\ &Null[\partial^{(4)}_{0,j,0}] = 0, && Rk[\partial^{(4)}_{0,j,0}] = 2. \end{aligned}$$

*Proof.* This can be proved in the same way as the preceding lemma, by the symmetry of $t$ and $s$. □

**Lemma 4.20.** *For* $(0,0,k)$*, with* $i \neq 0$ *the ranks and nullities of* $\partial^{(0)},\ldots,\partial^{(4)}$ *are given by*

$$\begin{aligned} &Null[\partial^{(0)}_{0,0,k}] = 5, && \\ &Null[\partial^{(1)}_{0,0,k}] = 13, && Rk[\partial^{(1)}_{0,0,k}] = 5, \\ &Null[\partial^{(2)}_{0,0,k}] = 10, && Rk[\partial^{(2)}_{0,0,k}] = 13, \\ &Null[\partial^{(3)}_{0,0,k}] = 3, && Rk[\partial^{(3)}_{0,0,k}] = 9, \\ &Null[\partial^{(4)}_{0,0,k}] = 0, && Rk[\partial^{(4)}_{0,0,k}] = 3. \end{aligned}$$

*Proof.* We reason as in the previous two lemmas. Let $r$ be fixed. We instruct the program of Appendix C to find $[\partial^{(r)}_{0,0,k}]_R(\mathbf{t},\mathbf{s},\mathbf{u})$, and then to evaluate it in $\mathbf{t}$=$\mathbf{s}$= 1. The entries of the resulting matrix $[\partial^{(r)}_{0,0,k}]_R(1,1,\mathbf{u})$ take values on $\mathbb{C}[\mathbf{u}]$. As before, we then instruct the program to find the Smith form of $[\partial^{(r)}_{0,0,k}]_R(1,1,\mathbf{u})$. For each $r$, the diagonal of this Smith form is shown below.

$$\begin{aligned}
diag\left(S\left([\partial^{(1)}_{0,0,k}]_R(1,1,\mathbf{u})\right)\right) &= (1,1,1,1,1),\\
diag\left(S\left([\partial^{(2)}_{0,0,k}]_R(1,1,\mathbf{u})\right)\right) &= (1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0),\\
diag\left(S\left([\partial^{(3)}_{0,0,k}]_R(1,1,\mathbf{u})\right)\right) &= (1,1,1,1,1,1,1,1,1,0,0,0),\\
diag\left(S\left([\partial^{(4)}_{0,0,k}]_R(1,1,\mathbf{u})\right)\right) &= (1,1,1).
\end{aligned}$$

Let us notice that all the entries appearing on the listed diagonals are equal to 1 or 0. Since

$$S\left([\partial^{(r)}_{0,0,k}]_R(1,1,\mu_k)\right) = S\left([\partial^{(r)}_{0,0,k}]_R(1,1,\mathbf{u})\right)|_{\mathbf{u}=\mu_k},$$

then,

$$\begin{aligned}
Rk\left([\partial^{(r)}_{0,0,k}]\right) &= Rk\left([\partial^{(r)}_{0,0,k}]_R(1,1,\mu_k)\right),\\
&= Rk\left(S\left([\partial^{(r)}_{0,0,k}]_R(1,1,\mu_k)\right)\right),\\
&= Rk\left(S\left([\partial^{(1)}_{0,0,k}]_R(1,1,\mathbf{u})\right)|_{\mathbf{u}=\mu_k}\right).
\end{aligned}$$

This implies that, for each $r$, the rank of the matrix $[\partial^{(r)}_{0,0,k}]$ is the number of non-zero entries of $diag\left(S\left([\partial^{(r)}_{0,0,k}]_R(1,1,\mathbf{u})\right)\right)$.

On the other hand,

$$\dim(M^{(r)}_{0,0,k}) = \left|B^{(r)}_{0,0,k}\right| = \left|B^{(r)}_{t,s,u}\cup B^{(r)}_{t,u}\cup B^{(r)}_{s,u}\cup B^{(r)}_{u}\right|.$$

Therefore, for $r$ equal to 0, 1, 2, 3 and 4, $\dim(M^{(r)}_{0,0,k})$ is equal to 5, 18, 23, 12 and 3 respectively. The nullities can be calculated from here using the Rank Theorem. $\square$

**Lemma 4.21.** *For $(i,j,0)$, with $i,j\neq 0$ the ranks and nullities of $\partial^{(0)},\ldots,\partial^{(4)}$ are given by*

$$\begin{aligned}
&Null[\partial^{(0)}_{i,j,0}] = 4,\\
&Null[\partial^{(1)}_{i,j,0}] = 13, \quad Rk[\partial^{(1)}_{i,j,0}] = 4,\\
&Null[\partial^{(2)}_{i,j,0}] = 10, \quad Rk[\partial^{(2)}_{i,j,0}] = 13,\\
&Null[\partial^{(3)}_{i,j,0}] = 2, \quad Rk[\partial^{(3)}_{i,j,0}] = 10,\\
&Null[\partial^{(4)}_{i,j,0}] = 0, \quad Rk[\partial^{(4)}_{i,j,0}] = 2.
\end{aligned}$$

*Proof.* Let $r$ be fixed. We first instruct the program of Appendix C to find $[\partial^{(r)}_{i,j,0}]_R(\mathbf{t},\mathbf{s},\mathbf{u})$, by eliminating from $[\partial^{(r)}]_R$ the adequate rows and columns, and then to evaluate it in u= 1. The entries of the resulting matrix $[\partial^{(r)}_{i,j,0}]_R(\mathbf{t},\mathbf{s},1)$ take values on the ring $\mathbb{C}[\mathbf{t},\mathbf{s}]$. Since this ring is not a principal ideal domain, the the Smith form of a matrix is not in general defined. However, for the matrices $[\partial^{(r)}_{i,j,0}]_R(\mathbf{t},\mathbf{s},1)$ in particular, the Smith form does exist. We use the program to calculate their Smith forms as before, which provide us the ranks.

On the other hand,

$$\dim(M^{(r)}_{i,j,0}) = \left|B^{(r)}_{i,j,0}\right| = \left|B^{(r)}_{t,s,u} \cup B^{(r)}_{t,s}\right|.$$

Therefore, for $r$ equal to 0, 1, 2, 3 and 4, $\dim(M^{(r)}_{i,j,0})$ is equal to 4, 17, 23, 12 and 2 respectively. The nullities can be calculated from here using the Rank Theorem. □

**Lemma 4.22.** *For $(i,0,k)$, with $i,k \neq 0$ the ranks and nullities of $\partial^{(0)},\ldots,\partial^{(4)}$ are given by*

$$\begin{array}{ll}
Null[\partial^{(0)}_{i,0,k}] = 3, & \\
Null[\partial^{(1)}_{i,0,k}] = 12, & Rk[\partial^{(1)}_{i,0,k}] = 3, \\
Null[\partial^{(2)}_{i,0,k}] = 10, & Rk[\partial^{(2)}_{i,0,k}] = 12, \\
Null[\partial^{(3)}_{i,0,k}] = 2, & Rk[\partial^{(3)}_{i,0,k}] = 10, \\
Null[\partial^{(4)}_{i,0,k}] = 0, & Rk[\partial^{(4)}_{i,0,k}] = 2.
\end{array}$$

*Proof.* Let $r$ be fixed. We first instruct the program of Appendix C to find $[\partial^{(r)}_{i,0,k}]_R(\mathbf{t},\mathbf{s},\mathbf{u})$, by eliminating from $[\partial^{(r)}]_R$ the adequate rows and columns, and then to evaluate it in s= 1. The entries of the resulting matrix $[\partial^{(r)}_{i,0,k}]_R(\mathbf{t},1,\mathbf{u})$ take values on the ring $\mathbb{C}[\mathbf{t},\mathbf{u}]$. As before, for the matrices $[\partial^{(r)}_{i,0,k}]_R(\mathbf{t},1,\mathbf{u})$ in particular, the Smith form does exist. We use the program to calculate their Smith forms as before, which provide us the ranks.

On the other hand,

$$\dim(M^{(r)}_{i,0,k}) = \left|B^{(r)}_{i,0,k}\right| = \left|B^{(r)}_{t,s,u} \cup B^{(r)}_{t,u}\right|.$$

Therefore, for $r$ equal to 0, 1, 2, 3 and 4, $\dim(M^{(r)}_{i,j,0})$ is equal to 3, 15, 22, 12 and 2 respectively. The nullities can be calculated from here using the Rank Theorem. □

**Lemma 4.23.** *For $(0,j,k)$, with $j,k \neq 0$ the ranks and nullities of $\partial^{(0)},\ldots,\partial^{(4)}$ are given by*

$$\begin{array}{ll}
Null[\partial^{(0)}_{0,j,k}] = 3, & \\
Null[\partial^{(1)}_{0,j,k}] = 12, & Rk[\partial^{(1)}_{0,j,k}] = 3, \\
Null[\partial^{(2)}_{0,j,k}] = 10, & Rk[\partial^{(2)}_{0,j,k}] = 12, \\
Null[\partial^{(3)}_{0,j,k}] = 2, & Rk[\partial^{(3)}_{0,j,k}] = 10, \\
Null[\partial^{(4)}_{0,j,k}] = 0, & Rk[\partial^{(4)}_{0,j,k}] = 2.
\end{array}$$

*Proof.* This can be proved in the same way as the preceding lemma, by the symmetry of $t$ and $s$. □

**Lemma 4.24.** *For $(i,j,k)$, with $i,j,k \neq 0$ the ranks and nullities of $\partial^{(0)},\ldots,\partial^{(4)}$ are given by*

$$\begin{array}{ll} Null[\partial^{(0)}_{i,j,k}] = 1, & \\ Null[\partial^{(1)}_{i,j,k}] = 11, & Rk[\partial^{(1)}_{i,j,k}] = 1, \\ Null[\partial^{(3)}_{i,j,k}] = 2, & Rk[\partial^{(3)}_{i,j,k}] = 10, \\ Null[\partial^{(4)}_{i,j,k}] = 0, & Rk[\partial^{(4)}_{i,j,k}] = 2. \end{array}$$

*and*

$$\begin{array}{lll} Null[\partial^{(2)}_{i,j,k}] = 10, & Rk[\partial^{(2)}_{i,j,k}] = 11 & \textit{if} \;\; \zeta_1 \neq \xi_j, \\ Null[\partial^{(2)}_{i,j,k}] = 11, & Rk[\partial^{(2)}_{i,j,k}] = 10 & \textit{if} \;\; \zeta_i = \xi_j. \end{array}$$

*Proof.* We reason as in the preceding lemmas. In this case the rank of $[\partial^{(r)}_{i,j,k}]_R(\zeta_i,\xi_j,\mu_k)$ varies depending on whether $\zeta_1 \neq \xi_j$ or $\zeta_i = \xi_j$.

Since,

$$\dim(M^{(r)}_{i,j,k}) = \left|B^{(r)}_{i,j,k}\right| = \left|B^{(r)}_{t,s,u}\right|,$$

for $r$ equal to 0, 1, 2, 3 and 4, $\dim(M^{(r)}_{i,j,k})$ is equal to 1, 12, 21, 12 and 2 respectively. The nullities can be calculated from here using the Rank Theorem. □

Let us observe that by Lemmas 4.12 and 4.14, the main homology chain may be decomposed as *abn* subchains of the form

$$M^{(4)}_{i,j,k} \xrightarrow{\partial^{(4)}_{i,j,k}} M^{(3)}_{i,j,k} \xrightarrow{\partial^{(3)}_{i,j,k}} M^{(2)}_{i,j,k} \xrightarrow{\partial^{(2)}_{i,j,k}} M^{(1)}_{i,j,k} \xrightarrow{\partial^{(1)}_{i,j,k}} M^{(0)}_{i,j,k} \xrightarrow{\partial^{(0)}_{i,j,k}} 0.$$

Therefore, the previous lemmas can be interpreted as providing the dimensions of the spaces of chains, boundaries, cycles and homology spaces for each of these subchains, in terms of the values of $i$, $j$ and $k$. Now we are in a position to prove the main theorem.

*Proof of Theorem 4.1.* As we pointed out before, for each $r$, $Rk[\partial^{(r)}]$, $Null[\partial^{(r)}]$ and $Null[\partial^{(r)}] - Rk[\partial^{(r)}]$ are the dimensions of the $r$-rth space of boundaries, the $r$-rth space of cycles and the $r$-th homology space respectively. By Lemma 4.15,

$$Rk[\partial^{(r)}] = \sum_{i,j,k} Rk[\partial^{(r)}_{i,j,k}] \;\; \text{and} \;\; Null[\partial^{(r)}] = \sum_{i,j,k} Null[\partial^{(r)}_{i,j,k}].$$

Moreover,

$$Null[\partial^{(r)}] - Rk[\partial^{(r)}] = \sum_{i,j,k} Null[\partial^{(r)}_{i,j,k}] - Rk[\partial^{(r)}_{i,j,k}].$$

Therefore, by Lemmas 4.17 to 4.24,

$$\begin{aligned}
Null[\partial^{(0)}] - Rk[\partial^{(0)}] &= \sum_{0,0,0} 1 + \sum_{i,0,0}^{i\neq 0} 0 + \sum_{0,j,0}^{j\neq 0} 0 + \sum_{0,0,k}^{k\neq 0} 0 \\
&\quad + \sum_{i,j,0}^{i,j\neq 0} 0 + \sum_{i,0,k}^{i,k\neq 0} 0 + \sum_{0,j,k}^{j,k\neq 0} 0 + \sum_{i,j,k}^{i,j,k\neq 0} 0 \\
&= 1,
\end{aligned}$$

$$\begin{aligned}
Null[\partial^{(1)}] - Rk[\partial^{(1)}] &= \sum_{0,0,0} 0 + \sum_{i,0,0}^{i\neq 0} 0 + \sum_{0,j,0}^{j\neq 0} 0 + \sum_{0,0,k}^{k\neq 0} 0 + \sum_{i,j,0}^{i,j\neq 0} 0 \\
&\quad + \sum_{i,0,k}^{i,k\neq 0} 0 + \sum_{0,j,k}^{j,k\neq 0} 0 + \sum_{i,j,k}^{i,j,k\neq 0,\ \zeta_1\neq\xi_j} 0 + \sum_{i,j,k}^{i,j,k\neq 0,\ \zeta_1=\xi_j} 1 \\
&= (d-1)(n-1),
\end{aligned}$$

$$\begin{aligned}
Null[\partial^{(2)}] - Rk[\partial^{(2)}] &= \sum_{0,0,0} 0 + \sum_{i,0,0}^{i\neq 0} 0 + \sum_{0,j,0}^{j\neq 0} 0 + \sum_{0,0,k}^{k\neq 0} 1 + \sum_{i,j,0}^{i,j\neq 0} 0 \\
&\quad + \sum_{i,0,k}^{i,k\neq 0} 0 + \sum_{0,j,k}^{j,k\neq 0} 0 + \sum_{i,j,k}^{i,j,k\neq 0,\ \zeta_1\neq\xi_j} 0 + \sum_{i,j,k}^{i,j,k\neq 0,\ \zeta_1=\xi_j} 1 \\
&= n-1+(d-1)(n-1),
\end{aligned}$$

$$\begin{aligned}
Null[\partial^{(3)}] - Rk[\partial^{(3)}] &= \sum_{i,j,k} 0 \quad = 0, \\
Null[\partial^{(4)}] - Rk[\partial^{(4)}] &= \sum_{i,j,k} 0 \quad = 0.
\end{aligned}$$

This provides the result. □

# Chapter 5

# Other Invariants of the Milnor Fiber and Fibration

Let $f(x, y, z) = z^n - x^a y^b$ and $\mathcal{CF}$ be as in the two previous chapters. Our purpose now is to calculate several invariants for the Milnor fiber $\mathcal{CF}$, and for the Milnor fibration of $f$, for arbitrary $a$, $b$ and $n$. Specifically, we calculate the monodromy of the fibration, the fundamental group, and integral homology of the fiber.

## 5.1 Monodromy of the Milnor Fibration

Let $\rho : F \longrightarrow F$ be the monodromy of the Milnor fibration of $f$, which is well defined up to isotopy.

**Theorem 5.1.** An expression for the monodromy $\rho$ of the Milnor fibration of $f$ is

$$\rho = t \circ u.$$

Or equivalently

$$\rho(x, y, z) = (e^{i\frac{2\pi}{a}} x, y, e^{i\frac{2\pi}{n}} z).$$

*Proof.* For $0 \leq r \leq 1$, let $F_r$ denote the Milnor fiber given by

$$F_r := \left\{ (x, y, z) \mid z^n - x^a y^b = e^{i2\pi r} \right\}.$$

Then, $\{F_r\}$ is the set of fibers of the Milnor fibration of $f$ over the circumference. Additionally, let us define a family $\{\rho_r : F_0 \longrightarrow F_r\}$ of diffeomorphisms by

$$\rho_r(x, y, z) = (e^{i\frac{2\pi r}{a}}x, y, e^{i\frac{2\pi r}{n}}z).$$

To see that $\rho_r$ is well defined, let us observe that, for $0 \leq r \leq 1$,

$$\begin{aligned}(e^{i\frac{2\pi r}{n}}z)^n - (e^{i\frac{2\pi r}{a}}x)^a y^b &= e^{i2\pi r}z^n - e^{i2\pi r}x^a y^b \\ &= e^{i2\pi r}(z^n - x^a y^b).\end{aligned}$$

Hence, if $(z^n - x^a y^b) = 1$, we have that $(e^{i\frac{2\pi r}{n}}z)^n - (e^{i\frac{2\pi r}{a}}x)^a y^b = e^{i2\pi r}$. And reciprocally, if $(e^{i\frac{2\pi r}{n}}z)^n - (e^{i\frac{2\pi r}{a}}x)^a y^b = e^{i2\pi r}$, then $(z^n - x^a y^b) = 1$. Therefore, it holds that $(x, y, z) \in F_0$ if and only if $\rho_r(x, y, z) \in F_r$, which implies that $\rho_r$ is well defined for every $r$. Then, by definition, $\rho_1$ is the monodromy of the fibration. By observing that $\rho_1 = \rho$ the result is complete. $\square$

## 5.2 Fundamental Group of the Complement of the Curve $xy(xy - 1) = 0$

Our aim now is to calculate the fundamental group of the compact Milnor fiber $\mathcal{CF}$. In order to do this we shall calculate first the fundamental group of the complement in $\mathbb{C}^2$ of the curve $xy(xy - 1) = 0$. We do this by using the classical Zariski-van Kampen method, though it can also be done by calculating the fundamental group of the two-skeleton of the complex $\mathcal{D}(B)$ constructed in Section 3.1.

In the first place, we transform the curve $xy(xy-1) = 0$ into $(y^2-x^2)(y^2-x^2+1) = 0$ by a change of variable in order to get rid of the vertical line and the asymptotes. Let us call this curve $C$, and let $f : \mathbb{C}^2 \longrightarrow \mathbb{C}$ be the function defined by $f(x, y) = (y^2-x^2)(y^2-x^2+1)$.

Then, there are only three values of $x$, which are $-1$, $0$, and $1$, in which $f(x, y)$ has multiple roots, or with the notation of the first chapter,

$$\Delta = \{x \in \mathbb{C} \mid f(x, y) \text{ has multiple roots}\} = \{-1, 0, 1\}.$$

This means that $L_{x=-1}$, $L_{x=0}$, and $L_{x=1}$ are the only vertical lines intersecting $C$ in less than four points.

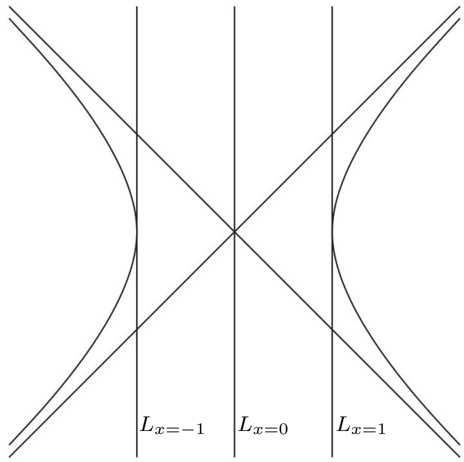


**Figure 5.1:** Real parts of $C$, $L_{x=-1}$, $L_{x=0}$, and $L_{x=1}$ in $\mathrm{Re}(\mathbb{C}) \times \mathrm{Re}(\mathbb{C})$.

Let us define

$$\begin{array}{rcl} C' & = & C \cup L_{x=-1} \cup L_{x=0} \cup L_{x=1}\,, \\ Z & = & \mathbb{C}^2 \setminus C'\,, \\ X & = & \mathbb{C} \setminus \{-1, 0, 1\}\,. \end{array}$$

Let us choose a base point $x_0 \in X$. For convenience, we choose $x_0$ to be equal to $1+\varepsilon$, for some $\varepsilon > 0$ to be defined. let $\phi : Z \longrightarrow X$ be the projection on the first coordinate, and let $F$ be defined by $F = \phi^{-1}(x_0)$, which is the line $L_{x=x_0}$ minus four points. Then $\phi$ is a locally trivial fiber bundle with fiber homeomorphic to $F$. Let us choose also a base point $y_0 \in F$. Then, we have the following exact sequence:

$$\pi_2(X, x_0) \longrightarrow \pi_1(F, y_0) \xrightarrow{\varphi} \pi_1(Z, (x_0, y_0)) \xrightarrow{\psi} \pi_1(X, x_0) \longrightarrow 1,$$

where $\varphi$ and $\psi$ are the homomorphisms induced by the inclusion of $F$ into $Z$ and the projection of $Z$ into $X$.

Let $\{\alpha_{-1}, \alpha_0, \alpha_1\}$ be a set of geometric generators for $\pi_1(X, x_0)$. For convenience, we choose each $\alpha_i$ to be as follows. Let $\varepsilon$ be a real number such that $0 < \varepsilon < 1/2$ and, for $j \in \{-1, 0, 1\}$, let $\gamma_j$ be the loop given by

$$\gamma_j(t) = j + \varepsilon e^{2i\pi t},$$

for $t \in [0, 1]$. Then we define $\alpha_1 = \gamma_1$ and, for $j \neq 1$, we define $\alpha_j$ as a composition of paths $\lambda_j \gamma_j \lambda_j^{-1}$, where $\lambda_j$ is some path joining $x_0$ with $j + \varepsilon$.

Let $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ be a set of geometric generators for $\pi_1(F, y_0)$. Then, the former exact sequence can be written as

$$1 \longrightarrow \langle \mu_1, \mu_2, \mu_3, \mu_4 \rangle \xrightarrow{\varphi} \pi_1(Z, (x_0, y_0)) \xrightarrow{\psi} \langle \alpha_{-1}, \alpha_0, \alpha_1 \rangle \longrightarrow 1.$$

We will use this exact sequence to calculate $\pi_1(Z,(x_0,y_0))$. For each $i$, let us denote $\varphi(\mu_i)$ by $\tilde{\mu}_i$. Also, let $\tilde{\alpha}_{-1}$, $\tilde{\alpha}_0$, and $\tilde{\alpha}_1$ be elements of $\pi_1(Z,(x_0,y_0))$ such that, for $i \in \{-1,0,1\}$, $\psi(\tilde{\alpha}_i) = \alpha_i$. We know that

$$\{\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3, \tilde{\mu}_4, \tilde{\alpha}_{-1}, \tilde{\alpha}_0, \tilde{\alpha}_1\}$$

is a set of generators for $\pi_1(Z,(x_0,y_0))$. Also, we know that $\pi_1(Z,(x_0,y_0))$ possesses the following relations, where the notation $w(g_1,\dots,g_n)$ means a word in the elements $g_1,\dots,g_n$.

- For each relation $w(\mu_1,\mu_2,\mu_3,\mu_4) = 1$ in $\pi_1(F,y_0)$, it is a trivial observation that $w(\tilde{\mu}_1,\tilde{\mu}_2,\tilde{\mu}_3,\tilde{\mu}_4) = 1$ is a relation in $\pi_1(Z,(x_0,y_0))$.
- For each relation $w(\alpha_{-1},\alpha_0,\alpha_1) = 1$ in $\pi_1(X,x_0)$ we have the following. It is clear that $\psi(w(\tilde{\alpha}_{-1},\tilde{\alpha}_0,\tilde{\alpha}_1)) = 1$, for which $w(\tilde{\alpha}_{-1},\tilde{\alpha}_0,\tilde{\alpha}_1) \in \ker(\psi) = im(\varphi)$. Therefore, there exists a word $w'(\mu_1,\mu_2,\mu_3,\mu_4)$ such that $\varphi(w'(\mu_1,\mu_2,\mu_3,\mu_4)) = w(\tilde{\alpha}_{-1},\tilde{\alpha}_0,\tilde{\alpha}_1)$. From here we obtain the relation $w(\tilde{\alpha}_{-1},\tilde{\alpha}_0,\tilde{\alpha}_1)w'^{-1}(\tilde{\mu}_1,\tilde{\mu}_2,\tilde{\mu}_3,\tilde{\mu}_4) = 1$.
- For each pair $(\mu_i,\alpha_j)$ we have the following. Since $\tilde{\mu}_i$ belongs to $im(\varphi) = \ker(\psi)$, which is a normal subgroup, then $\tilde{\alpha}_j^{-1}\tilde{\mu}_i\tilde{\alpha}_j$ belongs to $im(\varphi)$. From here it follows that, $\tilde{\alpha}_j^{-1}\tilde{\mu}_i\tilde{\alpha}_j = w_{i,j}(\tilde{\mu}_1,\tilde{\mu}_2,\tilde{\mu}_3,\tilde{\mu}_4)$ for some word $w_{i,j}$. We obtain in a similar way that $\tilde{\alpha}_j\tilde{\mu}_i\tilde{\alpha}_j^{-1} = w'_{i,j}(\tilde{\mu}_1,\tilde{\mu}_2,\tilde{\mu}_3,\tilde{\mu}_4)$, for some $w'_{i,j}$. These equalities provide two additional relations.

These relations are sufficient to determine the group $\pi_1(Z,(x_0,y_0))$. Moreover, since the groups $\pi_1(F,y_0)$ and $\pi_1(X,x_0)$ are free, we will only have relations of the third kind.

Before calculating these relations we will define an action

$$\cdot : \pi_1(X,x_0) \times \mathcal{B}_4 \longrightarrow \pi_1(X,x_0)$$

of the braid group $\mathcal{B}_4$ into the fundamental group $\pi_1(X,x_0)$ in the following way. First we define

$$\begin{array}{rcll} \mu_j \cdot \sigma_i & = & \mu_j & \text{if } j \neq i \text{ and } j \neq i+1, \\ \mu_j \cdot \sigma_i & = & \mu_{j+1} & \text{if } j = i, \\ \mu_j \cdot \sigma_i & = & \mu_{j+1}\mu_j\mu_{j+1}^{-1} & \text{if } j = i+1. \end{array}$$

The products of the form $\mu_j \cdot \sigma_i^{-1}$ are also given implicitly here. For an arbitrary loop $\gamma = \mu_{i_1} \cdot \ldots \cdot \mu_{i_r}$ we define $\gamma \cdot \sigma_i$ by

$$\gamma \cdot \sigma_i = (\mu_{i_1} \cdot \sigma_i) \cdot \ldots \cdot (\mu_{i_r} \cdot \sigma_i).$$

And for an arbitrary braid $b = \sigma_{i_1}^{\pm 1} \cdot \ldots \cdot \sigma_{i_r}^{\pm 1}$ we define $\gamma \cdot b$ by

$$\gamma \cdot b = (\gamma \cdot \sigma_{i_1}^{\pm 1}) \cdot \ldots \cdot (\gamma \cdot \sigma_{i_r}^{\pm 1}).$$

Geometrically, the image of a loop $\gamma$ by a braid $b$ can be obtained in the following way. Let us consider $b$ as a geometrical braid inside a cylinder, and let $B$ be the complement of $b$ in the cylinder. Let us see $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ as a geometrical set of generators for the bottom of $B$, which is a disk minus four points. By identifying the top and bottom of $B$, closing the braid, we can also see $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ at the top of $B$. Then, $\gamma \cdot b$ is obtained by taking the loop $\gamma$ at the bottom of $B$, and pushing it upwards, all the way to the top of $B$.

Let us also observe that, since $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ is a set of geometric generators, for every braid $b$ it holds that $(\tilde{\mu}_1 \cdots \tilde{\mu}_4) \cdot b = (\tilde{\mu}_1 \cdots \tilde{\mu}_4)$.

Now we return to our purpose of calculating the relations of the third type, i.e., of the form $\tilde{\alpha}_j^{-1}\tilde{\mu}_i\tilde{\alpha}_j = w_{i,j}(\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3, \tilde{\mu}_4)$, for $\pi_1(Z, (x_0, y_0))$. Let us observe that $\tilde{\alpha}_j^{-1}\tilde{\mu}_i\tilde{\alpha}_j$ is obtained by taking $\tilde{\mu}_i$ in $F$, and moving it along $\tilde{\alpha}_j$ all the way back to $F$. The word $w_{i,j}(\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3, \tilde{\mu}_4)$ expresses how $\tilde{\alpha}_j^{-1}\tilde{\mu}_i\tilde{\alpha}_j$ is to be read in terms of $\mu_1, \ldots, \mu_4$. From here we can see that, for every $i \in \{1, 2, 3, 4\}$ and $j \in \{-1, 0, 1\}$,

$$\tilde{\alpha}_j^{-1}\tilde{\mu}_i\tilde{\alpha}_j = \tilde{\mu}_i \cdot \rho(\alpha_j),$$

where

$$\rho : \pi_1(X, x_0) \longrightarrow \mathcal{B}_4$$

is the braid monodromy of $C$, presented as a homomorphism. Thus, for each $\tilde{\alpha}_j$ we have four relations associated to each $\tilde{\mu}_i$.

In order to calculate the relations we must therefore find a suitable presentation for $\rho$. For $i \in \{-1, 0, 1\}$, the following table shows the roots of $f(i, y)$ or, in other words, the $y$ values of the points at which $C$ intersect $L_{x=i}$.

$$\begin{array}{rcl}
1 & : & -x_0, \quad -\sqrt{x_0^2 - 1}, \quad \sqrt{x_0^2 - 1}, \quad x_0. \\
0 & : & -\varepsilon, \quad -\sqrt{\varepsilon^2 - 1}, \quad \sqrt{\varepsilon^2 - 1}, \quad \varepsilon. \\
-1 & : & -(1-\varepsilon), \quad -\sqrt{(1-\varepsilon)^2 - 1}, \quad \sqrt{(1-\varepsilon)^2 - 1}, \quad (1-\varepsilon) \ .
\end{array}$$

For $i \in \{-1, 0, 1\}$, let $\omega_i$ be the SCP in $i + \varepsilon$ that joins through straight segments the points of $L_{x=i}$ shown in the table, in the order they are listed. It can be directly calculated from the equation of $C$ that a representative of its braid monodromy is given by

$$\begin{aligned}
\rho(\alpha_1) &= \sigma_2, \\
\rho(\alpha_0) &= (\sigma_1^{-1}\sigma_3^{-1})\sigma_2^2(\sigma_1^{-1}\sigma_3^{-1})^{-1}, \\
\rho(\alpha_{-1}) &= (\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3\sigma_1)\sigma_2(\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3\sigma_1)^{-1},
\end{aligned}$$

where all the braids are defined according to $\omega_1$, $\omega_0$, and $\omega_{-1}$. Moreover, for $\varepsilon$ small enough, the braids $\sigma_2$, $\sigma_2^2$, and $\sigma_2$ can be taken as local braids associated respectively with $\gamma_1$, $\gamma_0$, and $\gamma_{-1}$, and the braids $\sigma_1^{-1}\sigma_3^{-1}$ and $\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_3\sigma_1$ as conjugating braids associated with $\lambda_0$ and $\lambda_{-1}$. Also, without loss of generality, we choose our geometric set of generators $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ to be coherent with $\omega_1$. This means that we define $\mu_1$ as a loop around $-x_0$, $\mu_2$ as a loop around $-\sqrt{x_0^2 - 1}$, and so on, in the order induced by $\omega_1$.

This allows us to calculate the relations explicitly. For the case of $\tilde{\alpha}_1$, the corresponding relations are given by

$$\tilde{\alpha}_1^{-1}\tilde{\mu}_i\tilde{\alpha}_1 = \tilde{\mu}_i \cdot \sigma_2.$$

Therefore, these relations are

$$\begin{array}{lrcl}
1. & \tilde{\alpha}_1^{-1}\tilde{\mu}_1\tilde{\alpha}_1 & = & \tilde{\mu}_1, \\
2. & \tilde{\alpha}_1^{-1}\tilde{\mu}_2\tilde{\alpha}_1 & = & \tilde{\mu}_3, \\
3. & \tilde{\alpha}_1^{-1}\tilde{\mu}_3\tilde{\alpha}_1 & = & \tilde{\mu}_3\tilde{\mu}_2\tilde{\mu}_3^{-1}, \\
4. & \tilde{\alpha}_1^{-1}\tilde{\mu}_4\tilde{\alpha}_1 & = & \tilde{\mu}_4.
\end{array}$$

Let us observe that, since $(\tilde{\mu}_1 \cdots \tilde{\mu}_4) \cdot \sigma_2 = (\tilde{\mu}_1 \cdots \tilde{\mu}_4)$, it also holds that

$$5. \qquad \tilde{\alpha}_1^{-1}(\tilde{\mu}_1 \cdots \tilde{\mu}_4)\tilde{\alpha}_1 = (\tilde{\mu}_1 \cdots \tilde{\mu}_4),$$

which provides us a fifth relation. Any of these five relations can be obtained from the other four, for which we can discard one of them. We choose to discard 3. Since 1, 4 and 5 are trivial relations they can be discarded too, for which the only relation we will keep is 2.

For the case of $\tilde{\alpha}_0$ we reason in a slightly different way. Let $\{\delta_1, \delta_2, \delta_3, \delta_4\}$ be a set of geometric generators for $\pi_1(\phi^{-1}(\varepsilon))$ obtained by sending $\{\mu_1, \mu_2, \mu_3, \mu_4\}$ into $L_{x=\varepsilon}$ by a homeomorphism from $L_{x=x_0}$ into $L_{x=\varepsilon}$ that sends $\omega_1$ into $\omega_0$. Then, for each $i$,

$$\delta_i = \mu_i \cdot \sigma_3\sigma_1.$$

Now, since the local braid around 0 is $\sigma_2^2$, the relations for $\tilde{\alpha}_0$, associated with $\{\delta_1, \delta_2, \delta_3, \delta_4\}$, are given by

$$\tilde{\alpha}_0^{-1}\delta_i\tilde{\alpha}_0 = \delta_i \cdot \sigma_2^2.$$

Therefore, these relations are

$$\begin{array}{lrcl}
1. & \tilde{\alpha}_0^{-1}\delta_1\tilde{\alpha}_0 & = & \delta_1, \\
2. & \tilde{\alpha}_0^{-1}\delta_2\tilde{\alpha}_0 & = & \delta_3\delta_2\delta_3^{-1}, \\
3. & \tilde{\alpha}_0^{-1}\delta_3\tilde{\alpha}_0 & = & \delta_3\delta_2\delta_3\delta_2^{-1}\delta_3^{-1}, \\
4. & \tilde{\alpha}_0^{-1}\delta_4\tilde{\alpha}_0 & = & \delta_4, \\
5. & \tilde{\alpha}_0^{-1}(\delta_1 \cdots \delta_4)\tilde{\alpha}_0 & = & (\delta_1 \cdots \delta_4).
\end{array}$$

As before, we may discard one of these relations, for which we discard 3. We discard also 1, 4, and 5, since they are trivial relations, keeping only 2. In order to obtain 2 in terms of $\{\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3, \tilde{\mu}_4\}$ we must substitute each $\delta_i$ with $\mu_i \cdot \sigma_3\sigma_1$. By doing so we obtain

$$\tilde{\alpha}_0^{-1}\tilde{\mu}_2\tilde{\mu}_1\tilde{\mu}_2^{-1}\tilde{\alpha}_0 = \tilde{\mu}_4\tilde{\mu}_2\tilde{\mu}_1\tilde{\mu}_2^{-1}\tilde{\mu}_4.$$

By a similar procedure, we obtain the relation

$$\tilde{\alpha}_{-1}^{-1}\tilde{\mu}_2\tilde{\alpha}_{-1} = (\tilde{\mu}_2\tilde{\mu}_1^{-1}\tilde{\mu}_2^{-1}\tilde{\mu}_4)\tilde{\mu}_3(\tilde{\mu}_2\tilde{\mu}_1^{-1}\tilde{\mu}_2^{-1}\tilde{\mu}_4)^{-1}$$

from the case of $\tilde{\alpha}_{-1}$.

Thus, the generators $\{\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3, \tilde{\mu}_4, \tilde{\alpha}_{-1}, \tilde{\alpha}_0, \tilde{\alpha}_1\}$, along with the three relations we have found, provide a presentation for $\pi_1(Z, (x_0, y_0))$. Now we are going to calculate $\pi_1(\mathbb{C}^2 \setminus C, (x_0, y_0))$. In order to do so, we recover $\mathbb{C}^2 \setminus C$ from $Z$, by reintroducing the lines $L_{x=-1}$, $L_{x=0}$, $L_{x=1}$.

Let us observe that $\gamma_{-1}$, $\gamma_0$, and $\gamma_1$ can be chosen is such a way that they bound a disk in $\mathbb{C}^2 \setminus C$, for which $\tilde{\alpha}_{-1}$, $\tilde{\alpha}_0$, and $\tilde{\alpha}_1$ are all trivial in $\mathbb{C}^2 \setminus C$. Let

$$i_* : \pi_1(Z, (x_0, y_0)) \longrightarrow \pi_1(\mathbb{C}^2 \setminus C, (x_0, y_0))$$

be the homomorphism induced by the inclusion of $Z$ into $\mathbb{C}^2 \setminus C$, then we can use van Kampen's Theorem to show that

$$\ker(i_*) = \langle \tilde{\alpha}_{-1}, \tilde{\alpha}_0, \tilde{\alpha}_1 \rangle .$$

This means that not only $\tilde{\alpha}_{-1}$, $\tilde{\alpha}_0$, and $\tilde{\alpha}_1$ become trivial by reintroducing the lines, but also that these loops, and their products, are the only trivial loops resulting from such procedure.

From here it follows that $\{\tilde{\mu}_1, \tilde{\mu}_2, \tilde{\mu}_3, \tilde{\mu}_4\}$ is a set of generators for $\pi_1(\mathbb{C}^2 \setminus C, (x_0, y_0))$, and that the relations

$$\begin{aligned} \tilde{\mu}_2 &= \tilde{\mu}_3, \\ \tilde{\mu}_2\tilde{\mu}_1\tilde{\mu}_2^{-1} &= \tilde{\mu}_4\tilde{\mu}_2\tilde{\mu}_1\tilde{\mu}_2^{-1}\tilde{\mu}_4, \\ \tilde{\mu}_2 &= (\tilde{\mu}_2\tilde{\mu}_1^{-1}\tilde{\mu}_2^{-1}\tilde{\mu}_4)\tilde{\mu}_3(\tilde{\mu}_2\tilde{\mu}_1^{-1}\tilde{\mu}_2^{-1}\tilde{\mu}_4)^{-1} \end{aligned}$$

determine the group. Let us observe that the last relation is equivalent to

$$\tilde{\mu}_2 = (\tilde{\mu}_1^{-1}\tilde{\mu}_2^{-1}\tilde{\mu}_4)\tilde{\mu}_2(\tilde{\mu}_1^{-1}\tilde{\mu}_2^{-1}\tilde{\mu}_4)^{-1}.$$

From here, and by making $\mu_i = \tilde{\mu}_i$ for every $i$, we obtain that

$$\pi_1(\mathbb{C}^2 \setminus C) = \left\langle \mu_1, \mu_2, \mu_4 : \left[\mu_2\mu_1\mu_2^{-1}, \mu_4\right] = \left[\mu_2, \mu_1^{-1}\mu_2^{-1}\mu_4\right] = 1 \right\rangle ,$$

where $[a, b]$ denotes the word $aba^{-1}b^{-1}$. By defining $\mu'_1 = \mu_2\mu_1\mu_2^{-1}$, we have that

$$\mu_1^{-1}\mu_2^{-1} = \mu_2^{-1}\mu_2\mu_1^{-1}\mu_2^{-1} = \mu_2^{-1}\mu_1'^{-1}.$$

Then, the second relation of the group can be rewritten as $\left[\mu_2, \mu_2^{-1}\mu_1'^{-1}\mu_4\right] = 1$, or equivalently as $\left[\mu_2, \mu_1'^{-1}\mu_4\right] = 1$. We have proved the following.

**Theorem 5.2.** The fundamental group of $\mathbb{C}^2 \setminus C$ is

$$\left\langle \mu'_1, \mu_2, \mu_4 : [\mu'_1, \mu_4] = \left[\mu_2, \mu_1'^{-1}\mu_4\right] = 1 \right\rangle .$$

Here $\mu_2$ is a meridian of the hyperbola, while $\mu'_1$ and $\mu_4$ are meridians of the asymptotes.

## 5.3 Fundamental Group of the Milnor Fiber

Now we will use the fundamental group calculated in the previous section to find the fundamental group of $\mathcal{CF}$ by using of covering theory. Along this section we will employ the notation from Chapter 3. Let us recall that

$$\begin{aligned}
\mathcal{H}_{1,1} &= \left\{(x,y)\in\mathbb{C}^2 \mid xy-1=0\right\},\\
\mathcal{H}_{a,b} &= \left\{(x,y)\in\mathbb{C}^2 \mid x^ay^b-1=0\right\},\\
B &= \left\{(x,y)\in\mathbb{C}^2 \mid \|x\|\le\varepsilon, \|y\|\le\varepsilon\right\},\\
B' &= \left\{(x,y)\in\mathbb{C}^2 \mid \|x\|\le\sqrt[a]{\varepsilon}, \|y\|\le\sqrt[b]{\varepsilon}\right\},
\end{aligned}$$

where $\varepsilon>1$. Let us recall also that

$$\begin{array}{c} P:\mathbb{C}^2\to\mathbb{C}^2\\ (x,y)\mapsto(x^a,y^b)\end{array} \quad\text{and}\quad \begin{array}{c} Q:\mathcal{F}\to\mathbb{C}^2\\ (x,y,z)\mapsto(x,y)\end{array}.$$

Let us define the following maps

$$\begin{array}{c} P_a:\mathbb{C}^2\to\mathbb{C}^2\\ (x,y)\mapsto(x^a,y)\end{array} \quad\text{and}\quad \begin{array}{c} P_b:\mathbb{C}^2\to\mathbb{C}^2\\ (x,y)\mapsto(x,y^b)\end{array}.$$

Lets us also define

$$\begin{aligned}
\mathcal{H}_{a,1} &= \left\{(x,y)\in\mathbb{C}^2 \mid x^ay-1=0\right\},\\
B_{1,1} &= B,\\
B_{a,1} &= \left\{(x,y)\in\mathbb{C}^2 \mid \|x\|\le\sqrt[a]{\varepsilon}, \|y\|\le\varepsilon\right\},\\
B_{a,b} &= B'.
\end{aligned}$$

For simplicity, we keep denoting the restrictions $P_a\mid_{B_{a,1}}$, $P_b\mid_{B_{a,b}}$, and $Q\mid_{\mathcal{CF}}$ by $P_a$, $P_b$, and $Q$. These maps are branched coverings of order $a$, $b$, and $n$ respectively. Finally, for $i\in\{1,a\}$ and $j\in\{1,b\}$ we define the sets

$$\begin{aligned}
X_{i,j} &= B_{i,j}\setminus\mathcal{H}_{i,j},\\
X_{i,j,y} &= B_{i,j}\setminus(\mathcal{H}_{i,j}\cup L_{y=0}),\\
X_{i,j,x,y} &= B_{i,j}\setminus(\mathcal{H}_{i,j}\cup L_{x=0}\cup L_{y=0}),\\
F_{a,b} &= \mathcal{CF}\setminus Q^{-1}(\mathcal{H}_{a,b})
\end{aligned}$$

(except for $(i,j)=(1,b)$), and the maps

$$\begin{aligned}
p_a &= P_a\mid_{X_{a,1,x,y}},\\
p_b &= P_b\mid_{X_{a,b,y}},\\
q &= Q\mid_{F_{a,b}}.
\end{aligned}$$

These maps are coverings of order $a$, $b$, and $n$ respectively. The situation is illustrated in the following commutative diagram, where all the arrows with hooks represent inclusion maps.

$$\begin{tikzcd}
 & & F_{a,b} \arrow[r, hook] \arrow[d, "q"] & \mathcal{CF} \arrow[d, "Q"] \\
 & X_{a,b,y} \arrow[r, hook] \arrow[d, "p_b"] & X_{a,b} \arrow[r, hook] & B_{a,b} \arrow[d, "P_b"] \\
X_{a,1,x,y} \arrow[r, hook] \arrow[d, "p_a"] & X_{a,1,y} \arrow[rr, hook] & & B_{a,1} \arrow[d, "P_b"] \\
X_{1,1,x,y} \arrow[rrr, hook] & & & B_{1,1}
\end{tikzcd}$$

Let us consider the maps

$$\begin{aligned}
\check{t} &\ : \ B_{a,1} \longrightarrow B_{a,1}, \\
\check{s} &\ : \ B_{a,b} \longrightarrow B_{a,b}, \text{ and} \\
t, s, u &\ : \ \mathcal{CF} \longrightarrow \mathcal{CF},
\end{aligned}$$

defined by

$$\begin{aligned}
\check{t}(x,y) &= (e^{\frac{2\pi}{a}} x, y), \\
\check{s}(x,y) &= (x, e^{\frac{2\pi}{b}} y), \\
t(x,y,z) &= (e^{\frac{2\pi}{a}} x, y, z), \\
s(x,y,z) &= (x, e^{\frac{2\pi}{b}} y, z), \text{ and} \\
u(x,y,z) &= (x, y, e^{\frac{2\pi}{n}} z).
\end{aligned}$$

Let us recall that $t$, $s$, and $u$ generate the deck transformations group of $Q \circ P_b \circ P_a$, which is therefore isomorphic to $\mathbb{Z}_a \oplus \mathbb{Z}_b \oplus \mathbb{Z}_n$. We also know that $\check{t}$, $\check{s}$, and $u$ generate the deck transformations groups of $P_a$, $P_b$, and $Q$ respectively. It follows from here that $\check{t}$, $\check{s}$, and $u$, restricted to the corresponding domains, also generate the deck transformations groups of $p_a$, $p_b$, and $q$ respectively. Therefore, these groups are $\mathbb{Z}_a$ for $p_a$, $\mathbb{Z}_b$ for $p_b$ and $\mathbb{Z}_n$ for $q$.

To find the fundamental group of $\mathcal{CF}$ we will successively calculate the fundamental groups of the spaces shown in the stairway-like part of the commutative diagram, from bottom to top, until reaching $\mathcal{CF}$.

By Theorem 5.2, we already know that

$$\pi_1(X_{1,1,x,y}) = \left\langle x, c, y : [x,y] = \left[c, xy^{-1}\right] = 1 \right\rangle,$$

where $x$, $c$, and $y$ are meridians of $L_{x=0}$, $\mathcal{H}_{1,1}$, and $L_{y=0}$ respectively. Then, our first aim is to calculate $\pi_1(X_{a,1,x,y})$. Let $\mu_a$ be the covering monodromy of $p_a$. Since $p_a$ is a cyclic covering, we know $\mu_a$ is given by

$$\begin{aligned} \mu_a : \pi_1(X_{1,1,x,y}) &\longrightarrow \Sigma_a. \\ c &\longmapsto 0 \\ y &\longmapsto 0 \\ x &\longmapsto (1,2,\ldots,a-1,a) \end{aligned}$$

The fact that $p_a$ is cyclic, implies also that it is regular (or normal), and therefore that

$$\pi_1(X_{a,1,x,y}) = \ker(\mu_a).$$

From here it follows that

$$\mu_a(\pi_1(X_{1,1,x,y})) = \frac{\pi_1(X_{1,1,x,y})}{\ker(\mu_a)}.$$

The regularity of $p_a$ implies that this quotient is isomorphic to its deck transformations group. Hence, we can write $\mu_a$ in the following way

$$\begin{aligned} \mu_a : \pi_1(X_{1,1,x,y}) &\longrightarrow \mathbb{Z}_a. \\ c &\longmapsto 0 \\ y &\longmapsto 0 \\ x &\longmapsto 1 \end{aligned}$$

We are now going to calculate $\ker(\mu_a)$. In general, and by definition, the CW complex associated with a group given by generators and relations consists of:

- A single vertex.
- An oriented edge for each generator. Each edge begins and finishes at the vertex.
- A disk for every relation. The boundary of each disk, as a sequence of edges, is given by the word that equals 1 in the corresponding relation.

Let us consider the CW complex associated with $\pi_1(X_{1,1,x,y})$, that we call $K_a$. This complex is defined by a vertex, three oriented edges that begin and finish at the vertex, that we name $x'$, $c'$, and $y'$, and two disks with boundaries $xyx^{-1}y^{-1}$ and $cxy^{-1}c^{-1}yx^{-1}$, that we name $D$ and $E$ respectively. It is clear that

$$\pi_1(K_a) = \pi_1(X_{1,1,x,y}).$$

We will construct a covering space $\tilde{K}_a$ of $K_a$, satisfying that the corresponding covering is regular and its monodromy is $\mu_a$. Then we will calculate the fundamental group of $\tilde{K}_a$.

To construct $\tilde{K}_a$ we consider an $a$-sided polygon, with edges oriented according to a given orientation of the circumference. We label its edges $x_0, \ldots, x_{a-1}$ in consecutive order. To each vertex of this polygon we attach two oriented loops. If the vertex is the initial point of some $x_i$, we label the loops $c_i$ and $y_i$. Finally, we add $2a$ disks, that we call $D_0, \ldots, D_{a-1}, E_0, \ldots E_{a-1}$, with boundaries given by

$$\begin{aligned} \partial D_i &= x_i y_{i+1} y_i^{-1} x_i^{-1} \quad \text{and} \\ \partial E_i &= c_i x_i y_{i+1}^{1} c_{i+1}^{-1} y_{i+1} x_i^{-1}, \end{aligned}$$

where $i+1$ is taken modulus $a$. The one-skeleton of $\tilde{K}_a$ is illustrated in the following figure. The disks $D_i$ (respectively $E_i$) can be easily imagined, for their boundaries start at the $i$-th vertex (the initial point of $x_i$), and from there read the word $xyx^{-1}y^{-1}$ (res. $cxy^{-1}c^{-1}yx^{-1}$) in the edges of $\tilde{K}_a$.

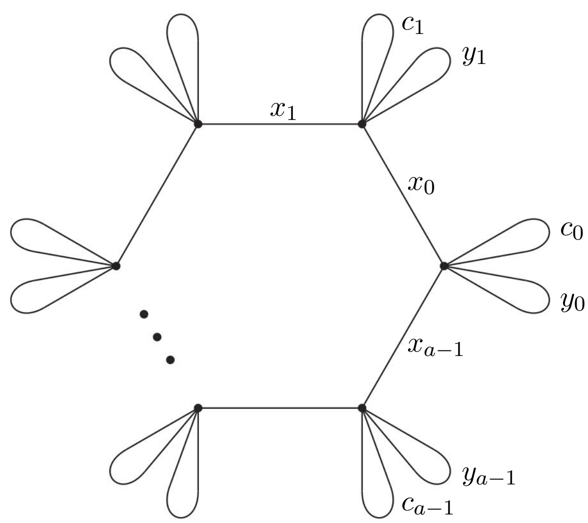


**Figure 5.2**

Now, let us consider the map that projects each $x_i$, $c_i$, and $y_i$ into $x'$, $c'$, and $y'$ respectively, respecting the orientations, each $D_i$ into $D$, and each $E_i$ into $E$. This map is a regular covering with monodromy equal to $\mu_a$, therefore,

$$\pi_1(\tilde{K}_a) = \ker(\mu_a) = \pi_1(X_{a,1,x,y}).$$

To calculate the fundamental group of $\tilde{K}_a$ we need to consider a maximal tree in its one-skeleton. we choose the tree $x_0, \ldots, x_{a-2}$, and choose the initial point of $x_0$ as a base point. By contracting this tree to a single point we obtain that $\pi_1(\tilde{K}_a)$ is generated by the remaining edges, i.e.

$$\{x_{a-1}, c_0, \ldots, c_{a-1}, y_0, \ldots, y_{a-1}\}.$$

The relations can be obtained from the boundaries of the disks, with the suppression of the $x_i$ for $i < a-1$. On the one hand, the $D_i$ provide the relations

$$
\text{(I).} \qquad \begin{array}{rcl}
y_1 y_0^{-1} &=& 1, \\
y_2 y_1^{-1} &=& 1, \\
&\vdots& \\
y_{a-1} y_{a-2}^{-1} &=& 1, \\
x_{a-1} y_{a-1} y_{a-2}^{-1} x_{a-1}^{-1} &=& 1.
\end{array}
$$

These relations imply that $y_i = y_j$ for every $i$ and $j$, which allows us to call $y_i$ simply by $y$. On the other hand, the $E_i$, provide the relations

$$
\text{(II).} \qquad \begin{array}{rcl}
c_0 y^1 c_1^{-1} y &=& 1, \\
c_1 y^1 c_2^{-1} y &=& 1, \\
&\vdots& \\
c_{a-2} y^1 c_{a-1}^{-1} y &=& 1, \\
c_{a-1} x_{a-1} y^1 c_0^{-1} y x_{a-1}^{-1} &=& 1.
\end{array}
$$

Thus we obtain that

$$
\pi_1(X_{a,1,x,y}) = \langle x_{a-1}, c_0, \ldots, c_{a-1}, y : [x_{a-1}, y] = 1,\ \text{(II)}\ \rangle\,.
$$

Now we calculate $\pi_1(X_{a,x,y})$. It is easy to see that the generator $x_{a-1}$ of $\pi_1(\tilde{K}_a)$ corresponds to a meridian of $L_{x=0}$ in $\pi_1(X_{a,1,x,y})$. This is true because they both correspond to the element $x^a$ of $\ker(\mu_a)$. Moreover, they are both the curve constructed from all the lifts of $x$ under the respective covering maps.

By reintroducing $L_{x=0}$ into $X_{a,1,x,y}$ we obtain, by the Seifert-van Kampen Theorem, that

$$
\pi_1(X_{a,1,y}) = \pi_1(X_{a,1,x,y}) \big/ \langle x_{a-1} \rangle\,.
$$

Therefore, by making $x_{a-1} = 1$ in the generators and relations of $\pi_1(X_{a,1,x,y})$, we obtain that $\pi_1(X_{a,x,y})$ is the group generated by

$$
\{c_0, \ldots, c_{a-1}, y\},
$$

with the relations

$$
\text{(III).} \qquad \begin{array}{rcl}
c_0 y^1 c_1^{-1} y &=& 1, \\
c_1 y^1 c_2^{-1} y &=& 1, \\
&\vdots& \\
c_{a-2} y^1 c_{a-1}^{-1} y &=& 1, \\
c_{a-1} y^1 c_0^{-1} y &=& 1.
\end{array}
$$

Or, in other words,

$$\pi_1(X_{a,1,y}) = \langle c_0, \ldots, c_{a-1}, y : \text{ (III) } \rangle .$$

This presentation can still be greatly simplified, until being reduced to a presentation with only two generators ($c$ and $y$) and one relation ($[c, y^a] = 1$). However, we will keep the big presentation on order to ease later calculations.

Now we shall repeat the whole procedure for $p_b$, in order to find $\pi_1(X_{a,b,y})$ and $\pi_1(X_{a,b})$. Let $\mu_b$ be the covering monodromy of $p_b$, which is given by

$$\begin{aligned} \mu_b : \pi_1(X_{a,1,y}) &\longrightarrow \Sigma_b. \\ c_0, \ldots, c_{a-1} &\longmapsto 0 \\ y &\longmapsto (1, 2, \ldots, b-1, b) \end{aligned}$$

As before, since $p_b$ is regular, we have that

$$\pi_1(X_{a,b,y}) = \ker(\mu_b).$$

From here it follows that

$$\mu_b(\pi_1(X_{a,1,y})) = \frac{\pi_1(X_{a,1,y})}{\ker(\mu_b)},$$

where the regularity of $p_b$ implies that this quotient is isomorphic to its deck transformations group. Hence, we can write $\mu_b$ in the following way

$$\begin{aligned} \mu_b : \pi_1(X_{a,1,y}) &\longrightarrow \mathbb{Z}_b. \\ c_0, \ldots, c_{a-1} &\longmapsto 0 \\ y &\longmapsto 1 \end{aligned}$$

We are now going to calculate $\ker(\mu_b)$. As before, we will consider the CW complex associated with $\pi_1(X_{a,1,y})$, that we call $K_b$. The complex possesses $a+1$ edges that we denote by $y', c'_0, \ldots, c'_{a-1}$. It also possesses $a$ disks with boundaries as given by the $a$ relations of (III). We denote these disks by $D_0, \ldots, D_{a-1}$, where $D_i$ is the disk associated with $c_i$ and $c_{i+1}$. Then we have that

$$\pi_1(K_b) = \pi_1(X_{a,1,y}).$$

As we did for $K_a$, we will construct a covering space $\tilde{K}_b$ of $K_b$, satisfying that the corresponding covering is regular and its monodromy is $\mu_b$. Then we will calculate the fundamental group of $\tilde{K}_b$.

We construct $\tilde{K}_b$ from a $b$-sided polygon, with edges oriented according to a given orientation of the circumference. We label these edges $y_0, \ldots, y_{b-1}$ in consecutive order. To each vertex of this polygon we attach $a$ oriented loops. We label the loops starting at the initial point of some $y_j$ as $c_{0,j}, \ldots, c_{a-1,j}$. Finally, we add $ab$ disks that we call $D_{i,j}$, for $0 \le i < a$ and $0 \le j < b$. The boundary of $D_{i,j}$ is given by

$$\partial D_{i,j} = c_{i,j} y_{j-1}^{-1} c_{i+1,j-1}^{-1} y_{j-1},$$

where $i+1$ is taken modulus $a$ and $j-1$ modulus $b$. The one-skeleton of $\tilde{K}_b$ is illustrated in the following figure. The boundary of $D_{i,j}$ starts at the $j$-th vertex (the initial point of $y_j$), and from there read the word $c_i y^1 c_{i+1}^{-1} y$ in the edges of $\tilde{K}_b$.

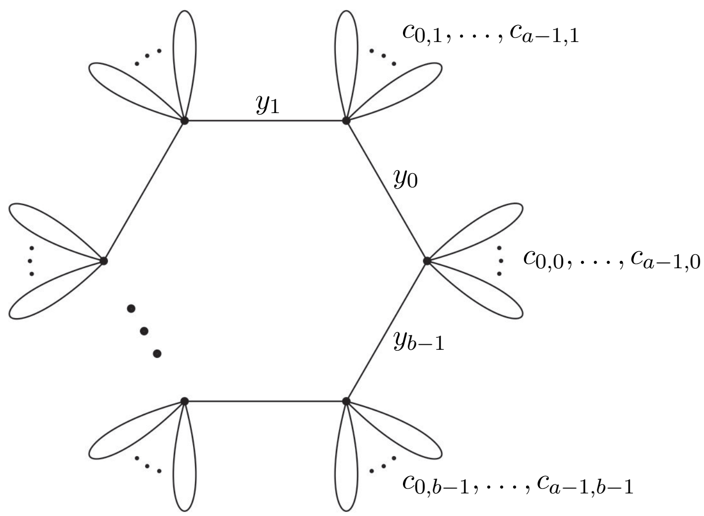


**Figure 5.3**

The map that projects each $c_{i,j}$ into $c'_i$, and each $y_j$ into $y'$, respecting the orientations, and each $D_{i,j}$ into $D_i$, is a regular covering with monodromy equal to $\mu_b$, therefore,

$$\pi_1(\tilde{K}_b) = \ker(\mu_b) = \pi_1(X_{a,b,y}).$$

To calculate the fundamental group of $\tilde{K}_b$ we choose the maximal tree $y_0, \ldots, y_{b-2}$, and the initial point of $y_0$ as a base point. By contracting this tree to a single point we obtain that $\pi_1(\tilde{K}_b)$ is generated by the remaining edges, i.e.

$$\{y_{b-1}\} \cup \{c_{0,j}, \ldots, c_{a-1,j}\}_{0 \leq j < b}.$$

The relations can be obtained from the boundaries of the disks, with the suppression of the $y_j$ for $i < b-1$. Then, for every $j$, the disks $D_{i,j}$, provide the relations

$$\text{(IV).} \qquad \begin{aligned} c_{0,j} c_{1,j-1}^{-1} &= 1, \\ c_{1,j} c_{2,j-1} &= 1, \\ &\vdots \\ c_{a-2,j} c_{a-1,j-1}^{-1} &= 1, \\ c_{a-1,j} y_{b-1}^{1} c_{0,j-1}^{-1} y_{b-1} &= 1. \end{aligned}$$

Thus we obtain $\pi_1(\tilde{K}_a)$.

Now we calculate $\pi_1(X_{a,b})$. As before, the generator $y_{b-1}$ of $\pi_1(\tilde{K}_b)$ corresponds to a meridian of $L_{y=0}$ in $\pi_1(X_{a,b,y})$, both corresponding to the element $y^b$ of $\ker(\mu_b)$. By reintroducing $L_{y=0}$ into $X_{a,b,y}$ we obtain, by the Seifert-van Kampen Theorem, that

$$\pi_1(X_{a,b}) = \pi_1(X_{a,b,y}) \diagup \langle y_{b-1} \rangle \,.$$

Therefore, $\pi_1(X_{a,b})$ is the group generated by

$$\{c_{0,j}, \ldots, c_{a-1,j}\}_{0<j<b-1},$$

with the relations

$$\begin{array}{rrcl}
(\mathrm{V}). & c_{0,j} & = & c_{1,j-1}, \\
 & c_{1,j} & = & c_{2,j-1}, \\
 & & \vdots & \\
 & c_{a-2,j} & = & c_{a-1,j-1}, \\
 & c_{a-1,j} & = & c_{0,j-1}^{-1}.
\end{array}$$

But this is exactly the free group generated by $c_{0,1}, \ldots, c_{d,1}$, where $d = \gcd(a, b)$. In other words,

$$\pi_1(X_{a,b}) = \langle c_0, \ldots, c_{d-1} \rangle \,.$$

It is important to observe that $c_0, \ldots, c_{d-1}$ are meridians of the $d$ irreducible components of $\mathcal{H}_{a,b}$.

Now, once again, we repeat the procedure for $q$, in order to find $\pi_1(F_{a,b})$ and $\pi_1(\mathcal{CF})$. Let $\mu_n$ be the covering monodromy of $q$, which is given by

$$\begin{aligned}
\mu_n : \pi_1(X_{a,b}) &\longrightarrow \Sigma_n. \\
c_0, \ldots, c_d &\longmapsto (1, 2, \ldots, n-1, n)
\end{aligned}$$

Once again, since $q$ is regular,

$$\pi_1(F_{a,b}) = \ker(\mu_n)$$

and

$$\begin{aligned}
\mu_n : \pi_1(X_{a,1,y}) &\longrightarrow \mathbb{Z}_b. \\
c_0, \ldots, c_{a-1} &\longmapsto 1
\end{aligned}$$

We are now going to calculate $\ker(\mu_n)$. For every $i$, with $0 \leq i < d$, let $t_i$ be defined by $t_i = c_0^{-1} c_i$. Then, $\mu_n(t_i) = 0$ for every $i$. Let $K_n$ be the CW complex associated with $\pi_1(X_{a,b})$, constructed by using the presentation $\langle c_0, t_1, \ldots, t_{d-1} \rangle$. We denote the edges of the complex by $c_0', t_1', \ldots, t_{d-1}'$. Since $K_n$ possesses no disks, $\pi_1(K_c)$ is free.

We will construct a covering space $\tilde{K}_n$ of $K_n$, such that the corresponding covering is regular and its monodromy is $\mu_n$. We construct $\tilde{K}_n$ from a $n$-sided polygon, with edges oriented according to a given orientation of the circumference. We label these edges $k_0, \ldots, k_{n-1}$ in consecutive order. To each vertex of this polygon we attach $d-1$ oriented loops. We label the loops starting at the initial point of $k_j$ as $t_{1,j}, \ldots, t_{d-1,j}$. The complex $\tilde{K}_b$ is illustrated in the following figure.

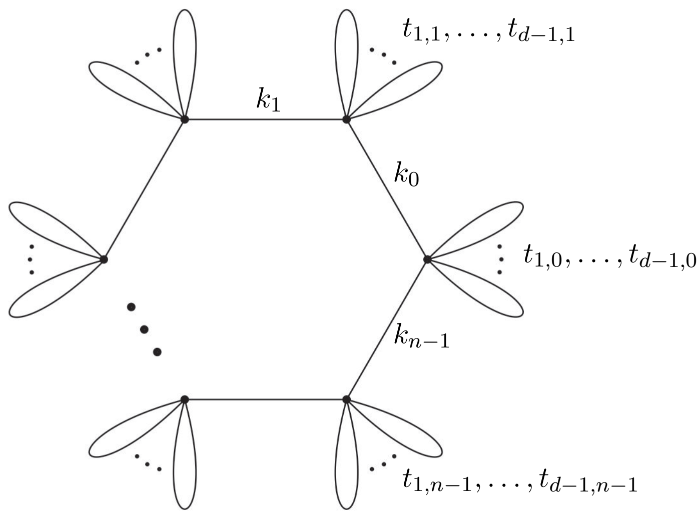


**Figure 5.4**

The map that projects each $t_{i,j}$ into $t'_i$, and each $k_j$ into $c'$, respecting the orientations, is a regular covering with monodromy equal to $\mu_n$, therefore,

$$\pi_1(\tilde{K}_n) = \ker(\mu_n) = \pi_1(F_{a,b}).$$

To calculate the fundamental group of $\tilde{K}_b$ we choose the maximal tree $k_0, \ldots, k_{n-2}$, and the initial point of $k_0$ as a base point. By contracting this tree to a single point we obtain that $\pi_1(\tilde{K}_n)$ is the free group generated by the remaining edges, i.e.

$$\{k_{n-1}\} \cup \{t_{1,j}, \ldots, t_{d-1,j}\}_{0 \leq j < n}.$$

It only remains to calculate $\pi_1(\mathcal{CF})$. Let us denote the irreducible components of $Q^{-1}(\mathcal{H}_{a,b})$ by $H_0, \ldots, H_{d-1}$. As before, the generator $k_{n-1}$ of $\pi_1(\tilde{K}_n)$ corresponds to a meridian of a branch of $Q^{-1}(\mathcal{H}_{a,b})$ in $\pi_1(F_{a,b})$, and is related to the element $c_{n-1}^n$ of $\ker(\mu_n)$. Let us assume that this branch is $H_0$. We know that this branch is an annulus, the fundamental group of which is generated by a single loop $l_0$. Besides, the intersection of a regular neighborhood of this branch and $F_{a,b}$ has the homotopy type or a torus, and its fundamental group is generated by two loops, which are homotopic to $l_0$ and $k_{n-1}$, provided a common base point. Since $k_{n-1}$ is a meridian of $H_0$, it is trivial in $F_{a,b} \cup H_0$. Besides, since the cone of $l_0$ in any of the copies of $B'$ that form $\mathcal{CF}$ is a disk that does not intersect $Q^{-1}(\mathcal{H}_{a,b})$, then $l_0$ is trivial in $F_{a,b}$.

Then, by reintroducing $H_0$ into $F_{a,b}$ we obtain, by the Seifert-van Kampen Theorem, that

$$\pi_1(F_{a,b} \cup H_0) = \frac{\pi_1(F_{a,b}) * \langle l_0 \rangle}{\langle k_{n-1}, l_0 \rangle} = \langle t_{1,j}, \ldots, t_{d-1,j} \rangle_{0 \leq j < n} .$$

On the other hand, let us observe that, for every $i$ with $0 < i < d$, the loop $k_{n-1}t_{i,0}, \ldots, k_{n-1}t_{i,n-1}$ of $\pi_1(\tilde{K}_n)$ is projected upon the loop $(c_{n-1}t_i)^n$ of $\pi_1(K_n)$. This loop is equal to $c_i^n$, by the definition of $t_i$. Let us recall also that $c_i$ is the meridian of a branch of $\mathcal{H}_{a,b}$ in $X_{a,b}$. Then, by reasoning as several times before, we see that $k_{n-1}t_{i,0}, \ldots, k_{n-1}t_{i,n-1}$ in $\pi_1(\tilde{K}_n)$ corresponds to a meridian of a branch $H_i$ in $\pi_1(F_{a,b})$, related to the element $c_i^n$ of $\ker(\mu_n)$. Let us observe also that the loop $k_{n-1}t_{i,0}, \ldots, k_{n-1}t_{i,n-1}$ in $\pi_1(F_{a,b})$ becomes the loop $t_{i,0}, \ldots, t_{i,n-1}$ in $\pi_1(F_{a,b} \cup H_0)$.

Then, by reintroducing $H_1$ into $F_{a,b} \cup H_0$ we obtain, by applying the Seifert-van Kampen Theorem in the same way as before, that

$$\begin{aligned} \pi_1(F_{a,b} \cup H_0 \cup H_1) &= \frac{\pi_1(F_{a,b} \cup H_0) * \langle l_1 \rangle}{\langle t_{1,0}, \ldots, t_{1,n-1}, l_1 \rangle} \\ &= \langle t_{1,j}, \ldots, t_{d-1,j} : t_{1,0}, \ldots, t_{1,n-1} = 1 \rangle_{0 \leq j < n} , \end{aligned}$$

where $l_1$ is the core of $H_1$. By repeating this procedure for each $H_i$ we obtain that the fundamental group of $\mathcal{CF}$ is the group generated by

$$\{t_{1,j}, \ldots, t_{d-1,j}\}_{0 \leq j < n},$$

and having the relations

$$\begin{aligned} t_{1,0}, \ldots, t_{1,n-1} &= 1, \\ &\vdots \\ t_{d-1,0}, \ldots, t_{d-1,n-1} &= 1. \end{aligned}$$

But these relations only mean that $t_{1,0}, \ldots, t_{d-1,0}$ can be defined in terms of the other generators, therefore

$$\pi_1(\mathcal{CF}) = \langle t_{1,j}, \ldots, t_{d-1,j} \rangle_{0 < j < n} .$$

We have proved the following.

**Theorem 5.3.** The fundamental group of $\mathcal{CF}$ is $\mathbb{F}_{(d-1)(n-1)}$, where $d := \gcd(a, b)$.

Here, $\mathbb{F}_{(d-1)(n-1)}$ denotes the free group generated by $(d-1)(n-1)$ elements.

## 5.4 Homology of the Milnor Fiber

Our aim now is to calculate the homology groups of $\mathcal{CF}$. These groups are given in the following theorem.

**Theorem 5.4.** The homology groups of $\mathcal{CF}$ are the following:

$$\begin{aligned} H_0(\mathcal{CF}) &= \mathbb{Z}, \\ H_1(\mathcal{CF}) &= \mathbb{Z}^{(d-1)(n-1)}, \\ H_2(\mathcal{CF}) &= \mathbb{Z}^{(d-1)(n-1)+n-1}, \\ H_3(\mathcal{CF}) &= 0, \\ H_4(\mathcal{CF}) &= 0, \end{aligned}$$

where $d := \gcd(a, b)$.

*Proof.* Since $\mathcal{CF}$ is path-connected, we know that $H_0(\mathcal{CF}) = \mathbb{Z}$. Also, since the Milnor fiber $\mathcal{F}$ is an affine algebraic variety of complex dimension two, we know that $\mathcal{F}$, and therefore $\mathcal{CF}$, has the homotopy type of a two-dimensional CW complex, for which

$$H_3(\mathcal{CF}) = H_4(\mathcal{CF}) = 0.$$

On the other hand, by Theorem 5.3, we know that $\pi_1(\mathcal{CF}) = \mathbb{F}_{(d-1)(n-1)}$. Therefore,

$$H_1(\mathcal{CF}) = \mathbb{Z}^{(d-1)(n-1)}.$$

It only remains to calculate the second homology group.

Let $b_i$ denote the $i$-th Betti number of $\mathcal{CF}$, i.e., the rank of $H_i(\mathcal{CF})$. Also, for any field $K$, let $b_{i;K}$ denote the dimension of $H_i(\mathcal{CF}; K)$. By the Fundamental Theorem of Finitely Generated Abelian Groups, $H_2(\mathcal{CF})$ has the form

$$H_2(\mathcal{CF}) = \mathbb{Z}^{b_2} \oplus \mathbb{Z}_{p_1^{a_1}} \oplus \cdots \oplus \mathbb{Z}_{p_m^{a_m}},$$

where, for each $i$, $p_i$ is a prime and $a_i$ a natural number. Besides, there is only one way to represent $H_2(\mathcal{CF})$ as a decomposition of this type. For every prime $p$, and every natural number $a$ let us define $k_{p,a}$ as the number of $\mathbb{Z}_{p^a}$ summands in $H_2(\mathcal{CF})$.

Let $p$ be a fixed prime number. Since all the homology groups of $\mathcal{CF}$ are finitely generated, the Universal Coefficient Theorem for Homology implies that, for every natural $m$, $H_m(\mathcal{CF}; \mathbb{Z}_p)$ is a direct sum with exactly the following summands:

- A $\mathbb{Z}_p$ summand for each $\mathbb{Z}$ summand in $H_m(\mathcal{CF})$.
- A $\mathbb{Z}_p$ summand for each summand in $H_m(\mathcal{CF})$ of the form $\mathbb{Z}_{p^a}$, with $a \geq 1$.

- A $\mathbb{Z}_p$ summand for each summand in $H_{m-1}(\mathcal{CF})$ of the form $\mathbb{Z}_{p^a}$, with $a \geq 1$.

(See [15, p. 264-266]) From here, and since since $H_3(\mathcal{CF}) = 0$, we have that

$$H_3(\mathcal{CF}; \mathbb{Z}_p) = \left( \sum_{a \geq 1} k_{p,a} \right) \mathbb{Z}_p,$$

where we use $c\mathbb{Z}_p$ as an alternative notation for $\mathbb{Z}_p^c$. However, since $\mathcal{CF}$ has the homotopy type of a two-dimensional complex, we know that

$$H_3(\mathcal{CF}; \mathbb{Z}_p) = 0.$$

From here it follows that for every prime $p$, and every natural number $a$, $k_{p,a} = 0$. Therefore

$$H_2(\mathcal{CF}) = \mathbb{Z}^{b_2}.$$

On the other hand, again by the Universal Coefficient Theorem for Homology, we know that

$$H_2(\mathcal{CF}) \otimes \mathbb{C} = H_2(\mathcal{CF}; \mathbb{C}),$$

which implies that,

$$b_2 = b_{2;\mathbb{C}}.$$

Therefore, by Theorem 4.1,

$$H_2(\mathcal{CF}) = \mathbb{Z}^{(d-1)(n-1)+n-1}.$$

□

# Bibliography


[1] Arnol'd, S.M., Gusein-Zade, S.M., Varchenko, A. N., *Singularities of differentiable maps*, Brikhäuser Boston Inc., Boston, MA, 1988.

[2] Artal, E., *Sur les couples de Zariski*, J. Algebraic Geom. **3** (1994), 223-247.

[3] Artal, E., Carmona, J., Cogolludo, J. I., *Braid monodromy and topology of plane curves*, Duke. Math J. **118** (2003), no. 2, 261-278.

[4] Artin, E., *Theory of braids*, Abh. Math. Sem. Hamburgischen Univ. **4** (1926), 47-72.

[5] Artin, E., *Theory of braids*, Ann. of Math. (2) **48** (1947), 101-126.

[6] Artin, E., *Braids and permutations*, Ann. of Math. (2) **48** (1947), 643-649.

[7] Ban, C., McEwan, L.J., Némethi, A., *On the Milnor fiber of a quasi-ordinary surface singularity*, Canad. J. Math. **54** (2002), 55-70.

[8] Birman, E., *Braids, links and mapping class groups*, Annals of Mathematics Studies, vol. 82, Princeton University Press, Princeton, N.J., 1974.

[9] Carmona, J., *Monodromía de trenzas de curvas algebraicas planas*, Tesis, Universidad de Zaragoza, 2003.

[10] Chisini, O., *Una suggestiva rappresentazione reale per le curve algebriche piane*, Ist. Lombardo, Rend., II. Ser. **66** (1933), 1141-1155.

[11] Enriques, F., *Sulla costruzione delle funzioni algebriche di due variabili possedenti una data curva di diramazione*, Annali Mat. Pura Appl. **I** (1924), 185-198.

[12] Fernández de Bobadilla J., Marco M. A., *Topology of hypersurface singularities with 3-dimensional critical set*, Comment. Math. Helv. **88** (2013), no. 2, 253-304.

[13] González, P. D., McEwan, L.J., Némethi, A., *The zeta-function of a quasi-ordinary singularity II*, Contemp. Math. **324** (2003), 109-122.

[14] Griffiths, P.A., *Introduction to Algebraic Curves*, Translations of Mathematical Monographs, vol. 76, AMS, Providence, R.I., 1989.

[15] Hatcher, A., *Algebraic Topology*, Cambridge University Press, Cambridge, 2002.

[16] van Kampen, E. R., *On the fundamental group of an algebraic curve*, Amer. J. Math. **55** (1933), 255-260.

[17] Kato, M., Matsumoto, Y., *On the connectivity of the Milnor fiber of a holomorphic function at a critical point*, Manifolds Tokyo 1973, University of Tokyo Press 1975, 131-136.

[18] Kulikov, V. S., Teicher, M., *Braid monodromy factorizations and diffeomorphism types*, Izv. Ross. Akad. Nauk Ser. Mat. **64** (2000), no. 2, 89-120.

[19] Libgober, A., *On the homotopy type of the complement to plane algebraic curves*, J. Reine Angew. Math. **367** (1986), 103-114.

[20] Lipman, J., *Topological invariants of quasi-ordinary singularities*, Mem. Amer. Math. Soc. **74** (1988), no. 388. 1-107.

[21] Michel, F., Pichon, A. *On the boundary of the Milnor fibre of non-isolated singularities*, Int. Math. Res. Not. **43** (2003), 2305-2311.

[22] Michel, F., Pichon, A. *Carrousel in family and non-isolatedhypersurface singularities in* $\mathbb{C}^3$, J. Reine Angew. Math. **720** (2016), 1-32.

[23] Michel, F., Pichon, A., Weber, C., *The boundary of the Milnor fiber of Hirzebruch surface singularities*, Singularity theory, 745-760, World Sci. Publ., Hackensack, NJ, 2007.

[24] Michel, F., Pichon, A., Weber, C., *The boundary of the Milnor fiber for some non-isolated singularities of complex surfaces*, Osaka J. Math. **46** (2009), 291-316.

[25] Milnor, J. W., *Singular points of complex hypersurfaces*, Annals of Mathematics Studies, vol. 61, Princeton University Press, Princeton, N.J., 1968.

[26] Moishezon, B. G., *Stable branch curves and braid monodromies*, Algebraic geometry (Chicago, III., 1980), Lecture Notes in Math., vol. 862, Springer, Berlin, 1981, pp. 107-192.

[27] Moishezon, B. G., Teicher, M, *Braid group technique in complex geometry I. Line arrangements in* $\mathbb{CP}^2$, Contemp. Math. **78** (1988), 425-555.

[28] Moishezon, B. G., Teicher, M, *Braid group technique in complex geometry II. From arrangements of lines and conics to cuspidal curves*, Algebraic Geometry, Lecture Notes in Math. **1479** (1990).

[29] Moishezon, B. G., Teicher, M, *Braid group technique in complex geometry III. Projective degeneration of $V_3$*, Contemp. Math. **162** (1992), 313-332.

[30] Moishezon, B. G., Teicher, M, *Braid group technique in complex geometry IV. Braid monodromy of the branched curve $S_3$ of $V_3 \longrightarrow \mathbb{CP}^2$ and application to $\pi_1(\mathbb{CP}^2 - S_3, *)$*, Contemp. Math. **162** (?), 332-358.

[31] Moishezon, B. G., Teicher, M, *Braid group technique in complex geometry V. The fundamental group of the complement of a branch curve of a Veronese generic projection*, Comm. Anal. Geom. **4** (1996), 1-120.

[32] Némethi, A., *The Milnor fiber and the zeta function of the singularities of type $f = P(h, g)$)*, Compositio Math. **79** (1991), 63-97.

[33] Némethi, A., *Hypersurface singularities with 2-dimensional critical locus*, J. London Math. Soc. **59** (1999), 922-938.

[34] Némethi, A., Szilárd, A., *Milnor fiber boundary of a non-isolated surface singularity*, Lecture Notes in Math., vol. 2037, Springer, Berlin, Heidelberg, 2012.

[35] Neumann, W.D., *A calculus for plumbing applied to the topology of complex surface singularities and degenerating complex curves*, Trans. Amer. Math. Soc. **268** (1981), no. 2, 299-344.

[36] Neuwirth, L. P., *Knot Groups*, Annals of Mathematics Studies, vol. 56, Princeton University Press, Princeton, N.J., 1965.

[37] Oka, M., *Symmetric plane curves with nodes and cusps*, J. Math. Soc. Japan **44** (1992), 375-414.

[38] Siersma, D., *Isolated line singularities*, Singularities, Part 2 (Arcata, Calif., 1981), 485-496, Proc. Sympos. Pure Math., **40**, Amer. Math. Soc., Providence, RI, 1983.

[39] Siersma, D., *Singularities with critical locus a 1-dimensional complete intersection and transversal type $A_1$*, Topol. Its Appl. **27** (1987), 51-73.

[40] Siersma, D., *Hypersurfaces with singular locus a plane curve and transversal type $A_1$*, Singularities (Warsaw, 1985), 397-410, Banach Center Publ., **20**, PWN, Warsaw, 1988.

[41] Siersma, D., *The vanishing topology of non isolated singularities*, New developments in singularity theory (Cambridge, 2000), 447-472, NATO Sci. Ser. II Math. Phys. Chem., **21**, Kluwer Acad. Publ., Dordrecht, 2001.

[42] Sigurdsson, B., *The Milnor fiber of the singularity* $f(x, y) + zg(x, y) = 0$, Rev. Mat. Complut. **29** (2016), no. 1, 225-239.

[43] Waldhausen, F., *Ein klasse von 3-dimensionalen mannigfaltigkeiten I, II,*, Invent. Math. 3 (1967), 308-333; ibid. **4** (1967), 87-117.

[44] Zariski, O., *On the problem of existence of algebraic functions of two variables possessing a given branch curve*, Amer. J. Math. **51** (1929), 305-328.

[45] Zariski, O., *The topological discriminant group of a Riemann surface of genus p*, Amer. J. Math. **59** (1937), 335-358.

[46] *SageMath, the Sage Mathematics Software System (Version 8.1)*, The Sage Developers, 2017, `https://www.sagemath.org`.

# Appendices

# *Appendix A*

# *Code for the CW Decomposition for Affine Plane Curves*

Here we exhibit the code of the program in SageMath that calculates the CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ from the braid monodromy of $\Omega$, where $\Omega$ is an affine plane curve and $\mathcal{D}$ a large enough polydisc. This program was explained in Section 2.1.

```
class _CaptureEq:
    '''
    Object wrapper that remembers "other" for successful equality tests.
    '''
    def __init__(self, obj):
        self.obj = obj
        self.match = obj
    def __eq__(self, other):
        result = (self.obj == other)
        if result:
            self.match = other
        return result
    def __getattr__(self, name):  # support hash() or anything else needed by __contains__
        return getattr(self.obj, name)

def get_equivalent(container, item, default=None):
    '''Gets the specific container element matched by: "item in container".
    Useful for retreiving a canonical value equivalent to "item".  For example, a
    caching or interning application may require fetching a single representative
    instance from many possible equivalent instances).

```

```
    >>> get_equivalent(set([1, 2, 3]), 2.0)             # 2.0 is equivalent to 2
    2
    >>> get_equivalent([1, 2, 3], 4, default=0)
    0
    '''
    t = _CaptureEq(item)
    if t in container:
        return t.match
    return default

class LocalBraid(object):
    def __init__(self, sing_point, braids, n, conj_braid):
        '''
        braids is the local braids list
        n[r] is the strands number of braids[r]
        '''
        self.sing_point = sing_point
        self.braids = braids
        l = max(len(b) for b in braids)
        for i in range(len(self.braids)):
            self.braids[i] = [0] + self.braids[i] + [0] * (l - len(self.braids[i]))
        self.n = n
        self.strands = sum(n)
        # k is the length of the components of beta
        self.k = len(self.braids[0])  # l+1
        self.conj_braid = conj_braid
        self.kc = len(self.conj_braid)

class BraidMonodromy(object):
    def __init__(self, local_braids):
        self.local_braids = local_braids
        strands = sum(self.local_braids[0].n)
        self.all_braids = [LocalBraid(0, [[0] * (len(self.local_braids) - 1)] * strands, [1] * strands,
                                      [])] + local_braids

    def CellularDescomposition(self):
        strands = sum(self.local_braids[0].n)
        base_tower = cells_of_tower(self.all_braids[0], self)
        lTower = [base_tower] + [cells_of_tower(local_braid, self) for local_braid in self.local_braids]
        lBridges = [cells_of_bridge(local_braid, self) for local_braid in self.local_braids]
        return join_cells(lTower + lBridges)

class Cell(object):
    def __str__(self):
        return self.name
    def __repr__(self):
        return self.name
```

```

class CellWithSign(object):
    def __init__(self,Cell,sgn):
        self.Cell = Cell
        self.sgn =sgn
    def __repr__(self):
        if self.sgn==1:
           return self.Cell.__repr__()
        else:
            return "-"+self.Cell.__repr__()
    def __eq__(self,other):
        return isinstance(other,CellWithSign) and (self.sgn==other.sgn) and
     (self.Cell==self.Cell)
    def __hash__(self):
        return hash((self.Cell,self.sgn))
    def cone(self):
        return CellWithSign(ConeCell(self.Cell),self.sgn)
    def product(self):
       if isinstance(self.Cell,BottomCell):
          return CellWithSign(product(self.Cell),self.sgn)
       else:
           raise Exception("A ProductCell must have a BottomCell as base")

class Chain(object):
    def __init__(self, set_of_cells_w_sign):
        self.d = {}
        for c in set_of_cells_w_sign:
            self.d[c.cell] = c.sgn

    def __add__(self, other):
        result = Chain(set({}))
        result.d = self.d.copy()
        for c in other.d:
            if c in result.d:
                coef = result.d[c] + other.d[c]
                if coef == 0:
                    del result.d[c]
                else:
                    result.d[c] = coef
            else:
                result.d[c] = other.d[c]
        return result

    def __iadd__(self, other):
        for c in other.d:
            if c in self.d:
                coef = self.d[c] + other.d[c]
                if coef == 0:
                    del self.d[c]
                else:
                    self.d[c] = coef
            else:
```

```
                self.d[c] = other.d[c]

        return self

    def __mul__(self, x):
        '''
        x: int
        '''
        result = Chain(set({}))
        result.d = self.d.copy()
        for c in result.d:
            result.d[c] = result.d[c] * x
        return result

    def degree(self):
        result = 0
        for c in self.d:
            result += self.d[c]
        return result

    def border(self):
        if len(self.d) != 0:
            dim = self.d.keys()[0].dim
            if dim == 0:
                return self.degree()
        result = Chain(set({}))
        for c in self.d:
            result += (Chain(c.border()) * self.d[c])
        return result

def cone(chain):
    """
    chain is a set of CellWithSign
    """
    return {e.cone() for e in chain}

def product_of_chain(border):
    """
    auxiliar function used in order to calculate the border of the
   cells_of_bridge
    """
    return {e.product() for e in border if isinstance(e.Cell,BottomCell)}

class TowerCell(Cell):
    def __init__(self,dim,name,index,sing_point,(i,r),mon):
        self.dim = dim
        self.name = name
        self.index = index
        self.sing_point = sing_point
        self.i = i
        self.r = r
        self.mon = mon
```

```python
    def __hash__(self):
        return hash((self.dim,self.name,self.index,self.sing_point,self.i,self.r))

    def __eq__(self,other):
        if not isinstance(other,TowerCell):
            return NotImplemented
        return isinstance(other,TowerCell) and (self.dim,self.name,self.index,self.sing_point,self.i,self.r)==(other.dim,other.name,other.index,other.sing_point,other.i,other.r)

    def border(self):
        beta = self.mon.all_braids[self.sing_point]
        if self.dim==0:
            return set({})
        elif self.name=="m2":
            c1 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(self.i,None),self.mon),1)
            c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(self.i,None),self.mon),-1)
            return {c1,c2}
        elif self.name=="m1":
            c1 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(self.i,None),self.mon),1)
            c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(self.i,None),self.mon),-1)
            return {c1,c2}
        elif self.name=="d":
            if self.index==0:
                c1 = CellWithSign(TowerCell(0,"A",1,self.sing_point,(self.i,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(self.i,None),self.mon),-1)
                return {c1,c2}
            elif self.index==beta.n[self.r-1]:
                c1 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(self.i,None),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2}
            elif self.index!=0 and self.index!=beta.n[self.r-1]:
                c1 = CellWithSign(TowerCell(0,"A",self.index+1,self.sing_point,(self.i,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2}
        elif self.name=="e":
            if self.index==0:
                c1 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(self.i % beta.k + 1,None),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(self.i,None),self.mon),-1)
```

```
                return {c1,c2}
            elif self.index==beta.strands+1:
                c1 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(self.i % beta.k + 1,None),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(self.i,None),self.mon),-1)
                return {c1,c2}
            elif self.index!=0 and self.index!=beta.strands+1:
                c1 = CellWithSign(TowerCell(0,"A",self.index,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2}
        elif self.name=="hq":
                q = abs(beta.braids[self.r-1][self.i-1])
                q = q - sum([beta.n[i] for i in xrange(self.r-1)])
                c1 = CellWithSign(TowerCell(0,"A",q+1,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",q,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2}
        elif self.name=="hq1":
                q = abs(beta.braids[self.r-1][self.i-1])
                q = q - sum([beta.n[i] for i in xrange(self.r-1)])
                c1 = CellWithSign(TowerCell(0,"A",q,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",q+1,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2}

        elif self.name=="lambda":
            result = {CellWithSign(TowerCell(1,"e",beta.strands+1,self.sing_point,(j,None),self.mon),1) for j in range(1,beta.k+1) }
            return result
        elif self.name=="mu":
            result = {CellWithSign(TowerCell(1,"e",0,self.sing_point,(j,None),self.mon),1) for j in range(1,beta.k+1) }
            return result
        elif self.name=="kappa":
            c1 = CellWithSign(TowerCell(1,"m1",None,self.sing_point,(self.i,self.r),self.mon),1)
            c2 = CellWithSign(TowerCell(1,"m1",None,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),-1)
            c3 = CellWithSign(TowerCell(1,"e",beta.strands+1,self.sing_point,(self.i,None),self.mon),-1)
            c4 = CellWithSign(TowerCell(1,"e",0,self.sing_point,(self.i,None),self.mon),1)
            return {c1,c2,c3,c4}
        elif self.name=="varkappa":
            c1 = CellWithSign(TowerCell(1,"m2",None,self.sing_point,(self.i,None),self.mon),1)
```

```
            c2 = CellWithSign(TowerCell(1,"m2",None,self.sing_point,(self.i % beta.k + 1,None),self.mon),-1)
            c3 = CellWithSign(TowerCell(1,"e",beta.strands+1,self.sing_point,(self.i,None),self.mon),-1)
            c4 = CellWithSign(TowerCell(1,"e",0,self.sing_point,(self.i,None),self.mon),1)
            return {c1,c2,c3,c4}
        elif self.name=="varsigma":
            q = abs(beta.braids[self.r-1][self.i-1])
            q = q - sum([beta.n[i] for i in xrange(self.r-1)])
            if self.index==0:
                c1 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                c3 = CellWithSign(TowerCell(1,"e",self.index,self.sing_point,(self.i,None),self.mon),1)
                c4 = CellWithSign(TowerCell(1,"e",self.index+1,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2,c3,c4}
            elif self.index==q:
                c1 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),-1)
                c2 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                c3 = CellWithSign(TowerCell(1,"hq",None,self.sing_point,(self.i,self.r),self.mon),1)
                c4 = CellWithSign(TowerCell(1,"hq1",None,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2,c3,c4}
            elif self.index==beta.n[self.r-1]:
                c1 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                c3 = CellWithSign(TowerCell(1,"e",self.index,self.sing_point,(self.i,self.r),self.mon),1)
                c4 = CellWithSign(TowerCell(1,"e",beta.strands+1,self.sing_point,(self.i,None),self.mon),-1)
                return {c1,c2,c3,c4}
            else:
                c1 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(1,"d",self.index,self.sing_point,(self.i,self.r),self.mon),-1)
                c3 = CellWithSign(TowerCell(1,"e",self.index,self.sing_point,(self.i,self.r),self.mon),1)
                c4 = CellWithSign(TowerCell(1,"e",self.index+1,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2,c3,c4}
        elif self.name=="theta":
```

```
            result = {CellWithSign(TowerCell(1,"d",j,self.sing_point,(self.i,self.r),self.mon),1) for j in range(0,beta.n[self.r-1]+1) }
            result.add(CellWithSign(TowerCell(1,"m1",None,self.sing_point,(self.i,self.r),self.mon),1))
            return result
        elif self.name=="vartheta":
            if self.r==len(beta.braids):
                result = {CellWithSign(TowerCell(1,"d",j,self.sing_point,(self.i,self.r),self.mon),-1) for j in range(0,beta.n[self.r-1]+1) }
                result.add(CellWithSign(TowerCell(1,"m2",None,self.sing_point,(self.i,None),self.mon),-1))
                return result
            else:
                result = {CellWithSign(TowerCell(1,"d",j,self.sing_point,(self.i,self.r),self.mon),-1) for j in range(0,beta.n[self.r-1]+1) }
                result.add(CellWithSign(TowerCell(1,"m1",None,self.sing_point,(self.i,self.r+1),self.mon),-1))
                return result
        elif self.name=="nu":
            q = abs(beta.braids[self.r-1][self.i-1])
            q = q - sum([beta.n[i] for i in xrange(self.r-1)])
            if self.index==1:
                c1 = CellWithSign(TowerCell(1,"d",q,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(1,"hq",None,self.sing_point,(self.i,self.r),self.mon),-1)
                c3 = CellWithSign(TowerCell(1,"e",q,self.sing_point,(self.i,self.r),self.mon),1)
                return {c1,c2,c3}
            elif self.index==2:
                c1 = CellWithSign(TowerCell(1,"d",q,self.sing_point,(self.i,self.r),self.mon),-1)
                c2 = CellWithSign(TowerCell(1,"hq",None,self.sing_point,(self.i,self.r),self.mon),1)
                c3 = CellWithSign(TowerCell(1,"e",q+1,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2,c3}
            elif self.index==3:
                c1 = CellWithSign(TowerCell(1,"d",q,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1)
                c2 = CellWithSign(TowerCell(1,"hq1",None,self.sing_point,(self.i,self.r),self.mon),1)
                c3 = CellWithSign(TowerCell(1,"e",q+1,self.sing_point,(self.i,self.r),self.mon),-1)
                return {c1,c2,c3}
            elif self.index==4:
                c1 = CellWithSign(TowerCell(1,"d",q,self.sing_point,(self.i,self.r),self.mon),-1)
                c2 = CellWithSign(TowerCell(1,"hq1",None,self.sing_point,(self.i,self.r),self.mon),-1)
                c3 = CellWithSign(TowerCell(1,"e",q,self.sing_point,(self.i,self.r),self.mon),1)
```

```
                return {c1,c2,c3}

        elif self.name=="Hsup":
            result = {CellWithSign(TowerCell(2,"varkappa",None,self.sing_point,(j,None),self.mon),-1) for j in range(1,beta.k+1) }
            result.add(CellWithSign(TowerCell(2,"mu",None,self.sing_point,(None,None),self.mon),1))
            result.add(CellWithSign(TowerCell(2,"lambda",None,self.sing_point,(None,None),self.mon),-1))
            return result
        elif self.name=="Hinf":
            result = {CellWithSign(TowerCell(2,"kappa",None,self.sing_point,(j,self.r),self.mon),-1) for j in range(1,beta.k+1) }
            result.add(CellWithSign(TowerCell(2,"mu",None,self.sing_point,(None,None),self.mon),1))
            result.add(CellWithSign(TowerCell(2,"lambda",None,self.sing_point,(None,None),self.mon),-1))
            return result
        elif self.name=="PI":
            s = sgn(beta.braids[self.r-1][self.i-1])
            t = abs(beta.braids[self.r-1][self.i-1])
            q = t - sum([beta.n[i] for i in xrange(self.r-1)])
            if t==0:
                result = {CellWithSign(TowerCell(2,"varsigma",j,self.sing_point,(self.i,self.r),self.mon),-1) for j in range(0,beta.n[self.r-1]+1)}
            else:
                result = {CellWithSign(TowerCell(2,"varsigma",j,self.sing_point,(self.i,self.r),self.mon),-1) for j in range(0,beta.n[self.r-1]+1) if j!=q}
                result.add(CellWithSign(TowerCell(2,"nu",2-s,self.sing_point,(self.i,self.r),self.mon),-1))
                result.add(CellWithSign(TowerCell(2,"nu",3-s,self.sing_point,(self.i,self.r),self.mon),-1))
            result.add(CellWithSign(TowerCell(2,"kappa",None,self.sing_point,(self.i,self.r),self.mon),1))
            result.add(CellWithSign(TowerCell(2,"theta",None,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1))
            result.add(CellWithSign(TowerCell(2,"theta",None,self.sing_point,(self.i,self.r),self.mon),-1))
            return result
        elif self.name=="OMEGA":
            s = sgn(beta.braids[self.r-1][self.i-1])
            t = abs(beta.braids[self.r-1][self.i-1])
            q = t - sum([beta.n[i] for i in xrange(self.r-1)])
            if t==0:
                result = {CellWithSign(TowerCell(2,"varsigma",j,self.sing_point,(self.i,self.r),self.mon),1) for j in range(0,beta.n[self.r-1]+1)}
            else:
                result = {CellWithSign(TowerCell(2,"varsigma",j,self.sing_point,(self.i,self.r),self.mon),1) for j in range(0,beta.n[self.r
```

```
-1]+1) if j!=q}
                result.add(CellWithSign(TowerCell(2,"nu",2+s,self.sing_point,(self.i,self.r),self.mon),1))
                result.add(CellWithSign(TowerCell(2,"nu",3+s,self.sing_point,(self.i,self.r),self.mon),1))
            result.add(CellWithSign(TowerCell(2,"vartheta",None,self.sing_point,(self.i % beta.k + 1,self.r),self.mon),1))
            result.add(CellWithSign(TowerCell(2,"vartheta",None,self.sing_point,(self.i,self.r),self.mon),-1))
            if self.r==len(beta.braids):
                result.add(CellWithSign(TowerCell(2,"varkappa",None,self.sing_point,(self.i,None),self.mon),-1))
            else:
                result.add(CellWithSign(TowerCell(2,"kappa",None,self.sing_point,(self.i,self.r+1),self.mon),-1))
            return result
        elif self.name=="PHIpi":
            s = sgn(beta.braids[self.r-1][self.i-1])
            q = abs(beta.braids[self.r-1][self.i-1])
            q = q - sum([beta.n[i] for i in xrange(self.r-1)])
            c1 = CellWithSign(TowerCell(2,"nu",1,self.sing_point,(self.i,self.r),self.mon),s)
            c2 = CellWithSign(TowerCell(2,"nu",4,self.sing_point,(self.i,self.r),self.mon),-s)
            c3 = CellWithSign(TowerCell(2,"varsigma",q,self.sing_point,(self.i,self.r),self.mon),s)
            return {c1,c2,c3}
        elif self.name=="PHIomega":
            s = sgn(beta.braids[self.r-1][self.i-1])
            q = abs(beta.braids[self.r-1][self.i-1])
            q = q - sum([beta.n[i] for i in xrange(self.r-1)])
            c1 = CellWithSign(TowerCell(2,"nu",2,self.sing_point,(self.i,self.r),self.mon),s)
            c2 = CellWithSign(TowerCell(2,"nu",3,self.sing_point,(self.i,self.r),self.mon),-s)
            c3 = CellWithSign(TowerCell(2,"varsigma",q,self.sing_point,(self.i,self.r),self.mon),-s)
            return {c1,c2,c3}

class ConeCell(Cell):
    def __init__(self,Cell):
        '''
        Cell is a TowerCell
        '''
        self.name="V("+Cell.name+")"
        self.dim=Cell.dim+1
        self.base=Cell
        self.sing_point=Cell.sing_point
        self.mon = Cell.mon
    def __eq__(self,other):
        if not isinstance(other, ConeCell):
            return NotImplemented
```

```
        return isinstance(other, ConeCell) and (self.name,self.dim,self.base)==(other.name,other.dim,other.base)
    def __hash__(self):
        return hash((self.base,self.name))
    def border(self):
        beta = self.mon.all_braids[self.sing_point]
        if self.base.name=="A":
            ####
            if self.base.index==0:
                c1 = CellWithSign(TowerCell(0,"AA",None,beta.sing_point,(None,self.base.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,beta.sing_point,(self.base.i,None),self.mon),-1)
                return {c1,c2}
            elif self.base.index==beta.strands+1:
                c1 = CellWithSign(TowerCell(0,"AA",None,beta.sing_point,(None,self.base.r),self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,beta.sing_point,(self.base.i,None),self.mon),-1)
                return {c1,c2}
            else:
                 return {CellWithSign(TowerCell(0,"AA",None,beta.sing_point,(None,self.base.r),self.mon),1),CellWithSign(self.base,-1)}
        elif self.base.name=="e":
            ####
            if self.base.index==0 or self.base.index==beta.strands+1:
                c = copy(self.base)
                c.r = None
                base_border = c.border()
                for Cell_sign in base_border:
                    Cell_sign.Cell.r=self.base.r
                result=cone(base_border)
                result.add(CellWithSign(c,-(-1)**self.base.dim))
                return result
            else:
                result=cone(self.base.border())
                result.add(CellWithSign(self.base,-(-1)**self.base.dim))
                return result
        elif self.base.name=="m1" or self.base.name=="d":
            c = copy(self.base)
            base_border = c.border()
            for Cell_sign in base_border:
                Cell_sign.Cell.r=self.base.r
            result=cone(base_border)
            result.add(CellWithSign(self.base,-(-1)**self.base.dim))
            return result
        elif self.base.name=="m2":
            ####
            if self.base.r!=None:
                c = copy(self.base)
                c.name = "m1"
                c.r = self.base.r +1
```

```
                base_border = c.border()
                for Cell_sign in base_border:
                    Cell_sign.Cell.r=self.base.r
                result=cone(base_border)
                result.add(CellWithSign(c,-(-1)**self.base.dim))
                return result
            else:
                c = copy(self.base)
                base_border = c.border()
                for Cell_sign in base_border:
                    Cell_sign.Cell.r=len(beta.braids)
                result=cone(base_border)
                result.add(CellWithSign(self.base,-(-1)**self.base.dim))
                return result
        elif self.base.name=="kappa" or self.base.name=="varsigma":
            c = copy(self.base)
            base_border = c.border()
            for Cell_sign in base_border:
                Cell_sign.Cell.r=self.base.r
            result=cone(base_border)
            result.add(CellWithSign(self.base,-(-1)**self.base.dim))
            return result
        elif self.base.name=="varkappa":
            ####
            if self.base.r!=None:
                c = copy(self.base)
                c.name = "kappa"
                c.r = self.base.r +1
                base_border = c.border()
                for Cell_sign in base_border:
                    Cell_sign.Cell.r=self.base.r
                    if Cell_sign.Cell.name=="m1":
                        Cell_sign.Cell.name = "m2"
                result=cone(base_border)
                result.add(CellWithSign(c,-(-1)**self.base.dim))
                return result
            else:
                c = copy(self.base)
                base_border = c.border()
                for Cell_sign in base_border:
                    if Cell_sign.Cell.name=="e":
                        Cell_sign.Cell.r=len(beta.braids)
                result=cone(base_border)
                result.add(CellWithSign(self.base,-(-1)**self.base.dim))
                return result
        elif self.base.name=="lambda" or self.base.name=="mu":
            ####
            c = copy(self.base)
            c.r = None
            base_border = c.border()
            for Cell_sign in base_border:
                Cell_sign.Cell.r=self.base.r
```

```
            result=cone(base_border)
            result.add(CellWithSign(c,-(-1)**self.base.dim))
            return result
        elif self.base.name=="vartheta":
            if self.base.r!=len(beta.braids):
                c = copy(self.base)
                base_border = c.border()
                for Cell_sign in base_border:
                    if Cell_sign.Cell.name=="m1":
                        Cell_sign.Cell.name = "m2"
                    Cell_sign.Cell.r=self.base.r
                result=cone(base_border)
                result.add(CellWithSign(c,-(-1)**self.base.dim))
                return result
            else:
                result=cone(self.base.border())
                result.add(CellWithSign(self.base,-(-1)**self.base.dim))
                return result
        elif self.base.name=="Hsup":
            ####
            if self.base.r!=None:
                c = copy(self.base)
                c.name = "Hinf"
                c.r = self.base.r +1
                base_border = c.border()
                for Cell_sign in base_border:
                    if Cell_sign.Cell.name=="kappa":
                        Cell_sign.Cell.name = "varkappa"
                    Cell_sign.Cell.r=self.base.r
                result=cone(base_border)
                result.add(CellWithSign(c,-(-1)**self.base.dim))
                return result
            else:
                c = copy(self.base)
                base_border = c.border()
                for Cell_sign in base_border:
                    if Cell_sign.Cell.name=="lambda" or Cell_sign.Cell.name
==" mu":
                        Cell_sign.Cell.r=len(beta.braids)
                result=cone(base_border)
                result.add(CellWithSign(self.base,-(-1)**self.base.dim))
                return result
        elif self.base.name=="Hinf":
            c = copy(self.base)
            base_border = c.border()
            for Cell_sign in base_border:
                Cell_sign.Cell.r=self.base.r
            result=cone(base_border)
            result.add(CellWithSign(self.base,-(-1)**self.base.dim))
            return result
        elif self.base.name=="OMEGA":
            if self.base.r!=len(beta.braids):
```

```
                c = copy(self.base)
                base_border = c.border()
                for Cell_sign in base_border:
                    if Cell_sign.Cell.name=="kappa":
                        Cell_sign.Cell.name = "varkappa"
                    Cell_sign.Cell.r=self.base.r
                result=cone(base_border)
                result.add(CellWithSign(c,-(-1)**self.base.dim))
                return result
            else:
                result=cone(self.base.border())
                result.add(CellWithSign(self.base,-(-1)**self.base.dim))
                return result
        else:
            result=cone(self.base.border())
            result.add(CellWithSign(self.base,-(-1)**self.base.dim))
            return result

class BottomCell(Cell):
    def __init__(self, dim, name,r , jr, sing_point, i, mon):
        self.dim = dim
        self.name = name
        self.r = r
        self.jr = jr
        self.sing_point=sing_point
        self.i = i
        self.mon = mon
    def __hash__(self):
        return hash((self.dim, self.name,self.r , self.jr, self.sing_point,
 self.i))
    def __eq__(self, other):
        if not isinstance(other, BottomCell):
            return NotImplemented
        return  isinstance(other, BottomCell) and (self.dim, self.name,
self.r , self.jr, self.sing_point, self.i) ==  (other.dim, other.name,
other.r , other.jr, other.sing_point, other.i)
    def border(self):
        beta = self.mon.all_braids[self.sing_point]
        n=sum(beta.n)
        l=len(beta.braids)
        kc = beta.kc

        #CELLS IN THE LOOM

        if self.dim==0:
            return set({})
        elif self.name=="m1":
            c1 = CellWithSign(BottomCell(0,"A",0,None,self.sing_point,self.
i,self.mon),1)
            c2 = CellWithSign(BottomCell(0,"A",beta.strands+1,None,self.
sing_point,self.i,self.mon),-1)
            return {c1,c2}
```

```
        elif self.name=="m2":
            c1 = CellWithSign(BottomCell(0,"A",0,None,self.sing_point,self.i,self.mon),1)
            c2 = CellWithSign(BottomCell(0,"A",beta.strands+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="d":
            c1 = CellWithSign(BottomCell(0,"A",self.r+1,None,self.sing_point,self.i,self.mon),1)
            c2 = CellWithSign(BottomCell(0,"A",self.r,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="e":
            c1 = CellWithSign(BottomCell(0,"A",self.r,None,self.sing_point,self.i+1,self.mon),1)
            c2 = CellWithSign(BottomCell(0,"A",self.r,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="hq":
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(BottomCell(0,"A",q+1,None,self.sing_point,self.i+1,self.mon),1)
            c2 = CellWithSign(BottomCell(0,"A",q,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="hq1":
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(BottomCell(0,"A",q,None,self.sing_point,self.i+1,self.mon),1)
            c2 = CellWithSign(BottomCell(0,"A",q+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}

        elif self.name=="theta":
            result = {CellWithSign(BottomCell(1,"d",j,None,self.sing_point,self.i,self.mon),1) for j in range(n+1) }
            result.add(CellWithSign(BottomCell(1,"m1",None,None,self.sing_point,self.i,self.mon),1))
            return result
        elif self.name=="vartheta":
            result = {CellWithSign(BottomCell(1,"d",j,None,self.sing_point,self.i,self.mon),-1) for j in range(n+1) }
            result.add(CellWithSign(BottomCell(1,"m2",None,None,self.sing_point,self.i,self.mon),-1))
            return result
        elif self.name=="kappa":
            c1 = CellWithSign(BottomCell(1,"m1",None,None,self.sing_point,self.i,self.mon),1)
            c2 = CellWithSign(BottomCell(1,"m1",None,None,self.sing_point,self.i+1,self.mon),-1)
            c3 = CellWithSign(BottomCell(1,"e",0,None,self.sing_point,self.i,self.mon),1)
```

```
            c4 = CellWithSign(BottomCell(1,"e",beta.strands+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2,c3,c4}
        elif self.name=="varkappa":
            c1 = CellWithSign(BottomCell(1,"m2",None,None,self.sing_point,self.i,self.mon),1)
            c2 = CellWithSign(BottomCell(1,"m2",None,None,self.sing_point,self.i+1,self.mon),-1)
            c3 = CellWithSign(BottomCell(1,"e",0,None,self.sing_point,self.i,self.mon),1)
            c4 = CellWithSign(BottomCell(1,"e",beta.strands+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2,c3,c4}
        elif self.name=="varsigma":
            if self.r==abs(beta.conj_braid[self.i-1]) and self.r != 0:
                c1 = CellWithSign(BottomCell(1,"hq",None,None,self.sing_point,self.i,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"hq1",None,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"d",self.r,None,self.sing_point,self.i+1,self.mon),-1)
                c4 = CellWithSign(BottomCell(1,"d",self.r,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3,c4}
            else:
                c1 = CellWithSign(BottomCell(1,"e",self.r,None,self.sing_point,self.i,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"e",self.r+1,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"d",self.r,None,self.sing_point,self.i+1,self.mon),1)
                c4 = CellWithSign(BottomCell(1,"d",self.r,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3,c4}
        elif self.name=="nu":
            q = abs(beta.conj_braid[self.i-1])
            if self.r==1:
                c1 = CellWithSign(BottomCell(1,"d",q,None,self.sing_point,self.i+1,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"hq",None,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"e",q,None,self.sing_point,self.i,self.mon),1)
                return {c1,c2,c3}
            if self.r==2:
                c1 = CellWithSign(BottomCell(1,"d",q,None,self.sing_point,self.i,self.mon),-1)
                c2 = CellWithSign(BottomCell(1,"hq",None,None,self.sing_point,self.i,self.mon),1)
                c3 = CellWithSign(BottomCell(1,"e",q+1,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3}
```

```
            if self.r==3:
                c1 = CellWithSign(BottomCell(1,"d",q,None,self.sing_point,self.i+1,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"hq1",None,None,self.sing_point,self.i,self.mon),1)
                c3 = CellWithSign(BottomCell(1,"e",q+1,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3}
            if self.r==4:
                c1 = CellWithSign(BottomCell(1,"d",q,None,self.sing_point,self.i,self.mon),-1)
                c2 = CellWithSign(BottomCell(1,"hq1",None,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"e",q,None,self.sing_point,self.i,self.mon),1)
                return {c1,c2,c3}

        elif self.name=="PI":
            s = sgn(beta.conj_braid[self.i-1])
            q = abs(beta.conj_braid[self.i-1])
            if q==0:
                result = {CellWithSign(BottomCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),-1) for j in range(0,beta.strands +1)}
            else:
                result = {CellWithSign(BottomCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),-1) for j in range(0,beta.strands +1) if j!=q}
                result.add(CellWithSign(BottomCell(2,"nu",2-s,None,self.sing_point,self.i,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"nu",3-s,None,self.sing_point,self.i,self.mon),-1))
            result.add(CellWithSign(BottomCell(2,"kappa",None,None,self.sing_point,self.i,self.mon),1))
            result.add(CellWithSign(BottomCell(2,"theta",None,None,self.sing_point,self.i+1,self.mon),1))
            result.add(CellWithSign(BottomCell(2,"theta",None,None,self.sing_point,self.i,self.mon),-1))
            return result
        elif self.name=="OMEGA":
            s = sgn(beta.conj_braid[self.i-1])
            q = abs(beta.conj_braid[self.i-1])
            if q==0:
                result = {CellWithSign(BottomCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),1) for j in range(0,beta.strands +1)}
            else:
                result = {CellWithSign(BottomCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),1) for j in range(0,beta.strands +1) if j!=q}
                result.add(CellWithSign(BottomCell(2,"nu",2+s,None,self.sing_point,self.i,self.mon),1))
                result.add(CellWithSign(BottomCell(2,"nu",3+s,None,self.sing_point,self.i,self.mon),1))
```

```
            result.add(CellWithSign(BottomCell(2,"varkappa",None,None,self.sing_point,self.i,self.mon),-1))
            result.add(CellWithSign(BottomCell(2,"vartheta",None,None,self.sing_point,self.i+1,self.mon),1))
            result.add(CellWithSign(BottomCell(2,"vartheta",None,None,self.sing_point,self.i,self.mon),-1))
            return result
        elif self.name=="PHIpi":
            s = sgn(beta.conj_braid[self.i-1])
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(BottomCell(2,"nu",1,None,self.sing_point,self.i,self.mon),s)
            c2 = CellWithSign(BottomCell(2,"nu",4,None,self.sing_point,self.i,self.mon),-s)
            c3 = CellWithSign(BottomCell(2,"varsigma",q,None,self.sing_point,self.i,self.mon),s)
            return {c1,c2,c3}
        elif self.name=="PHIomega":
            s = sgn(beta.conj_braid[self.i-1])
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(BottomCell(2,"nu",2,None,self.sing_point,self.i,self.mon),s)
            c2 = CellWithSign(BottomCell(2,"nu",3,None,self.sing_point,self.i,self.mon),-s)
            c3 = CellWithSign(BottomCell(2,"varsigma",q,None,self.sing_point,self.i,self.mon),-s)
            return {c1,c2,c3}

        # CELLS IN THE JOINTS

        elif self.name=="w0":
            if self.i==0:
                c1 = CellWithSign(BottomCell(0,"A",0,None,self.sing_point,1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,0,(self.sing_point,None),self.mon),-1)
                return {c1,c2}
            else:
                c1 = CellWithSign(BottomCell(0,"A",0,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(1,None),self.mon),-1)
                return {c1,c2}
        elif self.name=="w1":
            if self.i==0:
                c1 = CellWithSign(BottomCell(0,"A",beta.strands+1,None,self.sing_point,1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,0,(self.sing_point,None),self.mon),-1)
                return {c1,c2}
            else:
```

```
                c1 = CellWithSign(BottomCell(0,"A",beta.strands+1,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(1,None),self.mon),-1)
                return {c1,c2}
        elif self.name=="z":
            if self.i==0:
                c1 = CellWithSign(BottomCell(0,"A",self.r,None,self.sing_point,1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",1,0,(self.sing_point,self.r),self.mon),-1)
                return {c1,c2}
            else:
                S=sum([beta.n[i-1] for i in range(1,self.r)])
                c1 = CellWithSign(BottomCell(0,"A",S+self.jr,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",self.jr,self.sing_point,(1,self.r),self.mon),-1)
                return {c1,c2}
        elif self.name=="psi":
            if self.i==0:
                c1 = CellWithSign(BottomCell(1,"w0",None,None,self.sing_point,0,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"w1",None,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"m2",None,None,self.sing_point,1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m2",None,0,(self.sing_point,None),self.mon),1)
                return {c1,c2,c3,c4}
            else:
                c1 = CellWithSign(BottomCell(1,"w0",None,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"w1",None,None,self.sing_point,kc+1,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"m2",None,None,self.sing_point,kc+1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m2",None,self.sing_point,(1,None),self.mon),1)
                return {c1,c2,c3,c4}
        elif self.name=="xi":
            if self.i==0:
                c1 = CellWithSign(BottomCell(1,"w0",None,None,self.sing_point,0,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"w1",None,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"m1",None,None,self.sing_point,1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m1",None,0,(self.sing_point,1),self.mon),1)
                return {c1,c2,c3,c4}
            else:
```

```
                c1 = CellWithSign(BottomCell(1,"w0",None,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(BottomCell(1,"w1",None,None,self.sing_point,kc+1,self.mon),-1)
                c3 = CellWithSign(BottomCell(1,"m1",None,None,self.sing_point,kc+1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m1",None,self.sing_point,(1,1),self.mon),1)
                return {c1,c2,c3,c4}
        elif self.name=="zeta":
            S=sum([beta.n[i-1] for i in range(1,self.r)])
            c1 = CellWithSign(BottomCell(1,"z",self.r,self.jr,self.sing_point,kc+1,self.mon),1)
            c2 = CellWithSign(BottomCell(1,"z",self.r,self.jr+1,self.sing_point,kc+1,self.mon),-1)
            c3 = CellWithSign(BottomCell(1,"d",S+self.jr,None,self.sing_point,kc+1,self.mon),1)
            c4 = CellWithSign(TowerCell(1,"d",self.jr,self.sing_point,(1,self.r),self.mon),-1)
            return {c1,c2,c3,c4}
        elif self.name=="phi":
            if self.i==0:
                result = {CellWithSign(BottomCell(1,"d",j,None,self.sing_point,1,self.mon),1) for j in range(0,self.r)}
                result.add(CellWithSign(BottomCell(1,"w0",None,None,self.sing_point,0,self.mon),1))
                result.add(CellWithSign(BottomCell(1,"z",self.r,1,self.sing_point,0,self.mon),-1))
                result.add(CellWithSign(TowerCell(1,"d",0,0,(self.sing_point,self.r),self.mon),-1))
                return result
            else:
                S=sum([beta.n[i-1] for i in range(1,self.r)])
                result = {CellWithSign(BottomCell(1,"d",j,None,self.sing_point,kc+1,self.mon),1) for j in range(0,S+1)}
                result.add(CellWithSign(BottomCell(1,"w0",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(BottomCell(1,"z",self.r,1,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TowerCell(1,"d",0,self.sing_point,(1,self.r),self.mon),-1))
                return result
        elif self.name=="omega":
            if self.i==0:
                result = {CellWithSign(BottomCell(1,"d",j,None,self.sing_point,1,self.mon),1) for j in range(self.r,n+1)}
                result.add(CellWithSign(BottomCell(1,"w1",None,None,self.sing_point,0,self.mon),-1))
                result.add(CellWithSign(BottomCell(1,"z",self.r,1,self.sing_point,0,self.mon),1))
                result.add(CellWithSign(TowerCell(1,"d",1,0,(self.sing_point,self.r),self.mon),-1))
```

```
                return result
            else:
                S=sum([beta.n[i-1] for i in range(1,self.r+1)])
                result = {CellWithSign(BottomCell(1,"d",j,None,self.sing_point,kc+1,self.mon),1) for j in range(S,n+1)}
                result.add(CellWithSign(BottomCell(1,"w1",None,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(1,"z",self.r,beta.n[self.r-1],self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(1,"d",beta.n[self.r-1],self.sing_point,(1,self.r),self.mon),-1))
                return result
        elif self.name=="PSI":
            if self.i==0:
                c1 = CellWithSign(BottomCell(2,"phi",n,None,self.sing_point,0,self.mon),1)
                c2 = CellWithSign(BottomCell(2,"omega",n,None,self.sing_point,0,self.mon),1)
                c3 = CellWithSign(BottomCell(2,"psi",None,None,self.sing_point,0,self.mon),-1)
                c4 = CellWithSign(BottomCell(2,"vartheta",None,None,self.sing_point,1,self.mon),1)
                c5 = CellWithSign(TowerCell(2,"vartheta",None,0,(self.sing_point,n),self.mon),-1)
                return {c1,c2,c3,c4,c5}
            else:
                result = {CellWithSign(BottomCell(2,"zeta",l,j,self.sing_point,kc+1,self.mon),1) for j in range(1,beta.n[l-1])}
                result.add(CellWithSign(BottomCell(2,"phi",l,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(BottomCell(2,"omega",l,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(BottomCell(2,"psi",None,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"vartheta",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(2,"vartheta",None,self.sing_point,(1,l),self.mon),-1))
                return result
        elif self.name=="XI":
            if self.i==0:
                c1 = CellWithSign(BottomCell(2,"phi",1,None,self.sing_point,0,self.mon),-1)
                c2 = CellWithSign(BottomCell(2,"omega",1,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(BottomCell(2,"xi",None,None,self.sing_point,0,self.mon),1)
                c4 = CellWithSign(BottomCell(2,"theta",None,None,self.sing_point,1,self.mon),1)
                c5 = CellWithSign(TowerCell(2,"theta",None,0,(self.sing_point,1),self.mon),-1)
                return {c1,c2,c3,c4,c5}
```

```
            else:
                result = {CellWithSign(BottomCell(2,"zeta",1,j,self.sing_point,kc+1,self.mon),-1) for j in range(1,beta.n[0])}
                result.add(CellWithSign(BottomCell(2,"phi",1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"omega",1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"xi",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(BottomCell(2,"theta",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(2,"theta",None,self.sing_point,(1,1),self.mon),-1))
                return result
        elif self.name=="LAMDA":
            if self.i==0:
                c1 = CellWithSign(BottomCell(2,"phi",self.r+1,None,self.sing_point,0,self.mon),-1)
                c2 = CellWithSign(BottomCell(2,"omega",self.r+1,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(BottomCell(2,"phi",self.r,None,self.sing_point,0,self.mon),1)
                c4 = CellWithSign(BottomCell(2,"omega",self.r,None,self.sing_point,0,self.mon),1)
                c5 = CellWithSign(TowerCell(2,"theta",None,0,(self.sing_point,self.r+1),self.mon),-1)
                c6 = CellWithSign(TowerCell(2,"vartheta",None,0,(self.sing_point,self.r),self.mon),-1)
                return {c1,c2,c3,c4,c5,c6}
            else:
                result = {CellWithSign(BottomCell(2,"zeta",self.r,j,self.sing_point,kc+1,self.mon),1) for j in range(1,beta.n[self.r-1])}
                for j in range(1,beta.n[self.r]):
                    result.add(CellWithSign(BottomCell(2,"zeta",self.r+1,j,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"phi",self.r+1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"omega",self.r+1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(BottomCell(2,"phi",self.r,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(BottomCell(2,"omega",self.r,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(2,"theta",None,self.sing_point,(1,self.r+1),self.mon),-1))
                result.add(CellWithSign(TowerCell(2,"vartheta",None,self.sing_point,(1,self.r),self.mon),-1))
                return result

class TopCell(Cell):
    def __init__(self, dim, name,r , jr, sing_point, i,mon):
        self.dim = dim
```

```
        self.name = name
        self.r = r
        self.jr = jr
        self.sing_point=sing_point
        self.i = i
        self.mon = mon
    def __hash__(self):
        return hash((self.dim, self.name,self.r , self.jr, self.sing_point,
     self.i))
    def __eq__(self, other):
        if not isinstance(other, TopCell):
            return NotImplemented
        return   isinstance(other, TopCell) and (self.dim, self.name,self.r
     , self.jr, self.sing_point, self.i) ==  (other.dim, other.name,other.r
     , other.jr, other.sing_point, other.i)

    def border(self):
        beta = self.mon.all_braids[self.sing_point]
        n=sum(beta.n)
        l=len(beta.braids)
        kc = beta.kc

        # CELLS IN THE LOOM

        if self.dim==0:
            return set({})
        elif self.name=="m1":
            c1 = CellWithSign(TopCell(0,"A",0,None,self.sing_point,self.i,
    self.mon),1)
            c2 = CellWithSign(TopCell(0,"A",beta.strands+1,None,self.
    sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="m2":
            c1 = CellWithSign(TopCell(0,"A",0,None,self.sing_point,self.i,
    self.mon),1)
            c2 = CellWithSign(TopCell(0,"A",beta.strands+1,None,self.
    sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="d":
            c1 = CellWithSign(TopCell(0,"A",self.r+1,None,self.sing_point,
    self.i,self.mon),1)
            c2 = CellWithSign(TopCell(0,"A",self.r,None,self.sing_point,
    self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="e":
            c1 = CellWithSign(TopCell(0,"A",self.r,None,self.sing_point,
    self.i+1,self.mon),1)
            c2 = CellWithSign(TopCell(0,"A",self.r,None,self.sing_point,
    self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="hq":
            q = abs(beta.conj_braid[self.i-1])
```

```
            c1 = CellWithSign(TopCell(0,"A",q+1,None,self.sing_point,self.i+1,self.mon),1)
            c2 = CellWithSign(TopCell(0,"A",q,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}
        elif self.name=="hq1":
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(TopCell(0,"A",q,None,self.sing_point,self.i+1,self.mon),1)
            c2 = CellWithSign(TopCell(0,"A",q+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2}

        elif self.name=="theta":
            result = {CellWithSign(TopCell(1,"d",j,None,self.sing_point,self.i,self.mon),1) for j in range(n+1) }
            result.add(CellWithSign(TopCell(1,"m1",None,None,self.sing_point,self.i,self.mon),1))
            return result
        elif self.name=="vartheta":
            result = {CellWithSign(TopCell(1,"d",j,None,self.sing_point,self.i,self.mon),-1) for j in range(n+1) }
            result.add(CellWithSign(TopCell(1,"m2",None,None,self.sing_point,self.i,self.mon),-1))
            return result
        elif self.name=="kappa":
            c1 = CellWithSign(TopCell(1,"m1",None,None,self.sing_point,self.i,self.mon),1)
            c2 = CellWithSign(TopCell(1,"m1",None,None,self.sing_point,self.i+1,self.mon),-1)
            c3 = CellWithSign(TopCell(1,"e",0,None,self.sing_point,self.i,self.mon),1)
            c4 = CellWithSign(TopCell(1,"e",beta.strands+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2,c3,c4}
        elif self.name=="varkappa":
            c1 = CellWithSign(TopCell(1,"m2",None,None,self.sing_point,self.i,self.mon),1)
            c2 = CellWithSign(TopCell(1,"m2",None,None,self.sing_point,self.i+1,self.mon),-1)
            c3 = CellWithSign(TopCell(1,"e",0,None,self.sing_point,self.i,self.mon),1)
            c4 = CellWithSign(TopCell(1,"e",beta.strands+1,None,self.sing_point,self.i,self.mon),-1)
            return {c1,c2,c3,c4}
        elif self.name=="varsigma":
            if self.r==abs(beta.conj_braid[self.i-1]) and self.r != 0:
                c1 = CellWithSign(TopCell(1,"hq",None,None,self.sing_point,self.i,self.mon),1)
                c2 = CellWithSign(TopCell(1,"hq1",None,None,self.sing_point,self.i,self.mon),-1)
```

```
                c3 = CellWithSign(TopCell(1,"d",self.r,None,self.sing_point,self.i+1,self.mon),-1)
                c4 = CellWithSign(TopCell(1,"d",self.r,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3,c4}
            else:
                c1 = CellWithSign(TopCell(1,"e",self.r,None,self.sing_point,self.i,self.mon),1)
                c2 = CellWithSign(TopCell(1,"e",self.r+1,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"d",self.r,None,self.sing_point,self.i+1,self.mon),1)
                c4 = CellWithSign(TopCell(1,"d",self.r,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3,c4}
        elif self.name=="nu":
            q = abs(beta.conj_braid[self.i-1])
            if self.r==1:
                c1 = CellWithSign(TopCell(1,"d",q,None,self.sing_point,self.i+1,self.mon),1)
                c2 = CellWithSign(TopCell(1,"hq",None,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"e",q,None,self.sing_point,self.i,self.mon),1)
                return {c1,c2,c3}
            if self.r==2:
                c1 = CellWithSign(TopCell(1,"d",q,None,self.sing_point,self.i,self.mon),-1)
                c2 = CellWithSign(TopCell(1,"hq",None,None,self.sing_point,self.i,self.mon),1)
                c3 = CellWithSign(TopCell(1,"e",q+1,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3}
            if self.r==3:
                c1 = CellWithSign(TopCell(1,"d",q,None,self.sing_point,self.i+1,self.mon),1)
                c2 = CellWithSign(TopCell(1,"hq1",None,None,self.sing_point,self.i,self.mon),1)
                c3 = CellWithSign(TopCell(1,"e",q+1,None,self.sing_point,self.i,self.mon),-1)
                return {c1,c2,c3}
            if self.r==4:
                c1 = CellWithSign(TopCell(1,"d",q,None,self.sing_point,self.i,self.mon),-1)
                c2 = CellWithSign(TopCell(1,"hq1",None,None,self.sing_point,self.i,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"e",q,None,self.sing_point,self.i,self.mon),1)
                return {c1,c2,c3}

        elif self.name=="PI":
            s = sgn(beta.conj_braid[self.i-1])
```

```
            q = abs(beta.conj_braid[self.i-1])
            if q==0:
                result = {CellWithSign(TopCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),-1) for j in range(0,beta.strands+1)}
            else:
                result = {CellWithSign(TopCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),-1) for j in range(0,beta.strands+1) if j!=q}
                result.add(CellWithSign(TopCell(2,"nu",2-s,None,self.sing_point,self.i,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"nu",3-s,None,self.sing_point,self.i,self.mon),-1))
            result.add(CellWithSign(TopCell(2,"kappa",None,None,self.sing_point,self.i,self.mon),1))
            result.add(CellWithSign(TopCell(2,"theta",None,None,self.sing_point,self.i+1,self.mon),1))
            result.add(CellWithSign(TopCell(2,"theta",None,None,self.sing_point,self.i,self.mon),-1))
            return result
        elif self.name=="OMEGA":
            s = sgn(beta.conj_braid[self.i-1])
            q = abs(beta.conj_braid[self.i-1])
            if q==0:
                result = {CellWithSign(TopCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),1) for j in range(0,beta.strands+1)}
            else:
                result = {CellWithSign(TopCell(2,"varsigma",j,None,self.sing_point,self.i,self.mon),1) for j in range(0,beta.strands+1) if j!=q}
                result.add(CellWithSign(TopCell(2,"nu",2+s,None,self.sing_point,self.i,self.mon),1))
                result.add(CellWithSign(TopCell(2,"nu",3+s,None,self.sing_point,self.i,self.mon),1))
            result.add(CellWithSign(TopCell(2,"varkappa",None,None,self.sing_point,self.i,self.mon),-1))
            result.add(CellWithSign(TopCell(2,"vartheta",None,None,self.sing_point,self.i+1,self.mon),1))
            result.add(CellWithSign(TopCell(2,"vartheta",None,None,self.sing_point,self.i,self.mon),-1))
            return result
        elif self.name=="PHIpi":
            s = sgn(beta.conj_braid[self.i-1])
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(TopCell(2,"nu",1,None,self.sing_point,self.i,self.mon),s)
            c2 = CellWithSign(TopCell(2,"nu",4,None,self.sing_point,self.i,self.mon),-s)
            c3 = CellWithSign(TopCell(2,"varsigma",q,None,self.sing_point,self.i,self.mon),s)
            return {c1,c2,c3}
        elif self.name=="PHIomega":
            s = sgn(beta.conj_braid[self.i-1])
```

```
            q = abs(beta.conj_braid[self.i-1])
            c1 = CellWithSign(TopCell(2,"nu",2,None,self.sing_point,self.i,self.mon),s)
            c2 = CellWithSign(TopCell(2,"nu",3,None,self.sing_point,self.i,self.mon),-s)
            c3 = CellWithSign(TopCell(2,"varsigma",q,None,self.sing_point,self.i,self.mon),-s)
            return {c1,c2,c3}

        # CELLS IN THE JOINTS

        elif self.name=="w0":
            if self.i==0:
                c1 = CellWithSign(TopCell(0,"A",0,None,self.sing_point,1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,0,(self.sing_point % len(self.mon.local_braids) +1,None),self.mon),-1)
                return {c1,c2}
            else:
                c1 = CellWithSign(TopCell(0,"A",0,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",0,self.sing_point,(2,None),self.mon),-1)
                return {c1,c2}
        elif self.name=="w1":
            if self.i==0:
                c1 = CellWithSign(TopCell(0,"A",beta.strands+1,None,self.sing_point,1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,0,(self.sing_point % len(self.mon.local_braids) +1,None),self.mon),-1)
                return {c1,c2}
            else:
                c1 = CellWithSign(TopCell(0,"A",beta.strands+1,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",beta.strands+1,self.sing_point,(2,None),self.mon),-1)
                return {c1,c2}
        elif self.name=="z":
            if self.i==0:
                c1 = CellWithSign(TopCell(0,"A",self.r,None,self.sing_point,1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",1,0,(self.sing_point % len(self.mon.local_braids) +1,self.r),self.mon),-1)
                return {c1,c2}
            else:
                S=sum([beta.n[i-1] for i in range(1,self.r)])
                c1 = CellWithSign(TopCell(0,"A",S+self.jr,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TowerCell(0,"A",self.jr,self.sing_point,(2,self.r),self.mon),-1)
                return {c1,c2}
        elif self.name=="psi":
```

```
            if self.i==0:
                c1 = CellWithSign(TopCell(1,"w0",None,None,self.sing_point,0,self.mon),1)
                c2 = CellWithSign(TopCell(1,"w1",None,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"m2",None,None,self.sing_point,1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m2",None,0,(self.sing_point % len(self.mon.local_braids) +1,None),self.mon),1)
                return {c1,c2,c3,c4}
            else:
                c1 = CellWithSign(TopCell(1,"w0",None,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TopCell(1,"w1",None,None,self.sing_point,kc+1,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"m2",None,None,self.sing_point,kc+1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m2",None,self.sing_point,(2,None),self.mon),1)
                return {c1,c2,c3,c4}
        elif self.name=="xi":
            if self.i==0:
                c1 = CellWithSign(TopCell(1,"w0",None,None,self.sing_point,0,self.mon),1)
                c2 = CellWithSign(TopCell(1,"w1",None,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"m1",None,None,self.sing_point,1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m1",None,0,(self.sing_point % len(self.mon.local_braids) +1,1),self.mon),1)
                return {c1,c2,c3,c4}
            else:
                c1 = CellWithSign(TopCell(1,"w0",None,None,self.sing_point,kc+1,self.mon),1)
                c2 = CellWithSign(TopCell(1,"w1",None,None,self.sing_point,kc+1,self.mon),-1)
                c3 = CellWithSign(TopCell(1,"m1",None,None,self.sing_point,kc+1,self.mon),-1)
                c4 = CellWithSign(TowerCell(1,"m1",None,self.sing_point,(2,1),self.mon),1)
                return {c1,c2,c3,c4}
        elif self.name=="zeta":
            S=sum([beta.n[i-1] for i in range(1,self.r)])
            c1 = CellWithSign(TopCell(1,"z",self.r,self.jr,self.sing_point,kc+1,self.mon),1)
            c2 = CellWithSign(TopCell(1,"z",self.r,self.jr+1,self.sing_point,kc+1,self.mon),-1)
            c3 = CellWithSign(TopCell(1,"d",S+self.jr,None,self.sing_point,kc+1,self.mon),1)
            c4 = CellWithSign(TowerCell(1,"d",self.jr,self.sing_point,(2,self.r),self.mon),-1)
            return {c1,c2,c3,c4}
```

```
        elif self.name=="phi":
            if self.i==0:
                result = {CellWithSign(TopCell(1,"d",j,None,self.sing_point,1,self.mon),1) for j in range(0,self.r)}
                result.add(CellWithSign(TopCell(1,"w0",None,None,self.sing_point,0,self.mon),1))
                result.add(CellWithSign(TopCell(1,"z",self.r,1,self.sing_point,0,self.mon),-1))
                result.add(CellWithSign(TowerCell(1,"d",0,0,(self.sing_point % len(self.mon.local_braids) +1,self.r),self.mon),-1))
                return result
            else:
                S=sum([beta.n[i-1] for i in range(1,self.r)])
                result = {CellWithSign(TopCell(1,"d",j,None,self.sing_point,kc+1,self.mon),1) for j in range(0,S+1)}
                result.add(CellWithSign(TopCell(1,"w0",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TopCell(1,"z",self.r,1,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TowerCell(1,"d",0,self.sing_point,(2,self.r),self.mon),-1))
                return result
        elif self.name=="omega":
            if self.i==0:
                result = {CellWithSign(TopCell(1,"d",j,None,self.sing_point,1,self.mon),1) for j in range(self.r,n+1)}
                result.add(CellWithSign(TopCell(1,"w1",None,None,self.sing_point,0,self.mon),-1))
                result.add(CellWithSign(TopCell(1,"z",self.r,1,self.sing_point,0,self.mon),1))
                result.add(CellWithSign(TowerCell(1,"d",1,0,(self.sing_point % len(self.mon.local_braids) +1,self.r),self.mon),-1))
                return result
            else:
                S=sum([beta.n[i-1] for i in range(1,self.r+1)])
                result = {CellWithSign(TopCell(1,"d",j,None,self.sing_point,kc+1,self.mon),1) for j in range(S,n+1)}
                result.add(CellWithSign(TopCell(1,"w1",None,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(1,"z",self.r,beta.n[self.r-1],self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(1,"d",beta.n[self.r-1],self.sing_point,(2,self.r),self.mon),-1))
                return result
        elif self.name=="PSI":
            if self.i==0:
                c1 = CellWithSign(TopCell(2,"phi",n,None,self.sing_point,0,self.mon),1)
                c2 = CellWithSign(TopCell(2,"omega",n,None,self.sing_point,0,self.mon),1)
                c3 = CellWithSign(TopCell(2,"psi",None,None,self.sing_point,0,self.mon),-1)
```

```
                c4 = CellWithSign(TopCell(2,"vartheta",None,None,self.sing_point,1,self.mon),1)
                c5 = CellWithSign(TowerCell(2,"vartheta",None,0,(self.sing_point % len(self.mon.local_braids) +1,n),self.mon),-1)
                return {c1,c2,c3,c4,c5}
            else:
                result = {CellWithSign(TopCell(2,"zeta",l,j,self.sing_point,kc+1,self.mon),1) for j in range(1,beta.n[l-1])}
                result.add(CellWithSign(TopCell(2,"phi",l,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TopCell(2,"omega",l,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TopCell(2,"psi",None,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"vartheta",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(2,"vartheta",None,self.sing_point,(2,l),self.mon),-1))
                return result
        elif self.name=="XI":
            if self.i==0:
                c1 = CellWithSign(TopCell(2,"phi",1,None,self.sing_point,0,self.mon),-1)
                c2 = CellWithSign(TopCell(2,"omega",1,None,self.sing_point,0,self.mon),-1)
                c3 = CellWithSign(TopCell(2,"xi",None,None,self.sing_point,0,self.mon),1)
                c4 = CellWithSign(TopCell(2,"theta",None,None,self.sing_point,1,self.mon),1)
                c5 = CellWithSign(TowerCell(2,"theta",None,0,(self.sing_point % len(self.mon.local_braids) +1,1),self.mon),-1)
                return {c1,c2,c3,c4,c5}
            else:
                result = {CellWithSign(TopCell(2,"zeta",1,j,self.sing_point,kc+1,self.mon),-1) for j in range(1,beta.n[0])}
                result.add(CellWithSign(TopCell(2,"phi",1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"omega",1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"xi",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TopCell(2,"theta",None,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(2,"theta",None,self.sing_point,(2,1),self.mon),-1))
                return result
        elif self.name=="LAMDA":
            if self.i==0:
                c1 = CellWithSign(TopCell(2,"phi",self.r+1,None,self.sing_point,0,self.mon),-1)
                c2 = CellWithSign(TopCell(2,"omega",self.r+1,None,self.sing_point,0,self.mon),-1)
```

```
                c3 = CellWithSign(TopCell(2,"phi",self.r,None,self.sing_point,0,self.mon),1)
                c4 = CellWithSign(TopCell(2,"omega",self.r,None,self.sing_point,0,self.mon),1)
                c5 = CellWithSign(TowerCell(2,"theta",None,0,(self.sing_point % len(self.mon.local_braids) +1,self.r+1),self.mon),-1)
                c6 = CellWithSign(TowerCell(2,"vartheta",None,0,(self.sing_point % len(self.mon.local_braids) +1,self.r),self.mon),-1)
                return {c1,c2,c3,c4,c5,c6}
            else:
                result = {CellWithSign(TopCell(2,"zeta",self.r,j,self.sing_point,kc+1,self.mon),1) for j in range(1,beta.n[self.r-1])}
                for j in range(1,beta.n[self.r]):
                    result.add(CellWithSign(TopCell(2,"zeta",self.r+1,j,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"phi",self.r+1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"omega",self.r+1,None,self.sing_point,kc+1,self.mon),-1))
                result.add(CellWithSign(TopCell(2,"phi",self.r,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TopCell(2,"omega",self.r,None,self.sing_point,kc+1,self.mon),1))
                result.add(CellWithSign(TowerCell(2,"theta",None,self.sing_point,(2,self.r+1),self.mon),-1))
                result.add(CellWithSign(TowerCell(2,"vartheta",None,self.sing_point,(2,self.r),self.mon),-1))
                return result

class ProductCell(Cell):
    def __init__(self,  CellB, CellT):
        self.name = "I"+CellB.name
        self.dim = CellB.dim+1
        self.bottom = CellB
        self.top =CellT
        self.r = self.top.r
        self.jr = self.top.jr
        self.sing_point = self.top.sing_point
        self.i = self.top.i
        self.mon = CellB.mon
    def __eq__(self,other):
        if not isinstance(other,ProductCell):
            return NotImplemented
        return isinstance(other,ProductCell) and (self.bottom,self.top) == (other.bottom, other.top)
    def __hash__(self):
        return hash((self.bottom,self.top,self.name))
    def border(self):
        beta = self.mon.all_braids[self.sing_point]
        #ProductCells inherit all their indexes name, sing point, i, r.... from its associated TopCell and BottomCell
```

```
        if self.bottom.name in ["m2","m1","theta","vartheta","d","A","kappa","varkappa","PI","OMEGA","varsigma","e","hq","hq1","nu","PHIpi","PHIomega"]:
            if self.bottom.dim==0:
                return {CellWithSign(self.top,1),CellWithSign(self.bottom,-1)}
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                return result
        elif self.bottom.name == "w0":
            if self.bottom.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(1,"e",0,0,(self.sing_point,None),self.mon),-1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(1,"e",0,self.sing_point,(1,None),self.mon),-1))
                return result
        elif self.bottom.name == "w1":
            if self.bottom.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(1,"e",beta.strands+1,0,(self.sing_point,None),self.mon),-1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(1,"e",beta.strands+1,self.sing_point,(1,None),self.mon),-1))
                return result
        elif self.bottom.name == "z":
            if self.bottom.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
```

```
            result.add(CellWithSign(TowerCell(1,"e",self.jr,0,(self.sing_point,self.r),self.mon),-1))
            return result
        else:
            result = product_of_chain(self.bottom.border())
            result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
            result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
            result.add(CellWithSign(TowerCell(1,"e",self.jr,self.sing_point,(1,self.r),self.mon),-1))
            return result
    elif self.bottom.name == "psi":
        if self.bottom.i==0:
            result = product_of_chain(self.bottom.border())
            result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
            result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
            result.add(CellWithSign(TowerCell(2,"varkappa",None,0,(self.sing_point,None),self.mon),1))
            return result
        else:
            result = product_of_chain(self.bottom.border())
            result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
            result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
            result.add(CellWithSign(TowerCell(2,"varkappa",None,self.sing_point,(1,None),self.mon),1))
            return result
    elif self.bottom.name == "xi":
        if self.bottom.i==0:
            result = product_of_chain(self.bottom.border())
            result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
            result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
            result.add(CellWithSign(TowerCell(2,"kappa",None,0,(self.sing_point,1),self.mon),1))
            return result
        else:
            result = product_of_chain(self.bottom.border())
            result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
            result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
            result.add(CellWithSign(TowerCell(2,"kappa",None,self.sing_point,(1,1),self.mon),1))
            return result
    elif self.bottom.name == "phi":
        if self.bottom.i==0:
            result = product_of_chain(self.bottom.border())
            result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
            result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
```

```
                result.add(CellWithSign(TowerCell(2,"varsigma",0,0,(self.sing_point,self.r),self.mon),1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(2,"varsigma",0,self.sing_point,(1,self.r),self.mon),1))
                return result
        elif self.bottom.name == "omega":
            if self.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(2,"varsigma",1,0,(self.sing_point,self.r),self.mon),1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(2,"varsigma",beta.n[self.r-1],self.sing_point,(1,self.r),self.mon),1))
                return result
        elif self.bottom.name == "zeta":
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(2,"varsigma",self.jr,self.sing_point,(1,self.r),self.mon),1))
                return result
        elif self.bottom.name == "PSI":
            if self.bottom.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(3,"OMEGA",None,0,(self.sing_point,sum(beta.n)),self.mon),-1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(3,"OMEGA",None,self.sing_point,(1,len(beta.braids)),self.mon),-1))
```

```
                return result
        elif self.bottom.name == "XI":
            if self.bottom.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(3,"PI",None,0,(self.sing_point,1),self.mon),-1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(3,"PI",None,self.sing_point,(1,1),self.mon),-1))
                return result
        elif self.bottom.name == "LAMDA":
            if self.bottom.i==0:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(3,"OMEGA",None,0,(self.sing_point,self.r),self.mon),-1))
                result.add(CellWithSign(TowerCell(3,"PI",None,0,(self.sing_point,self.r+1),self.mon),-1))
                return result
            else:
                result = product_of_chain(self.bottom.border())
                result.add(CellWithSign(self.top,(-1)**self.bottom.dim))
                result.add(CellWithSign(self.bottom,-(-1)**self.bottom.dim))
                result.add(CellWithSign(TowerCell(3,"OMEGA",None,self.sing_point,(1,self.r),self.mon),-1))
                result.add(CellWithSign(TowerCell(3,"PI",None,self.sing_point,(1,self.r+1),self.mon),-1))
                return result


    def __hash__(self):
        return hash((self.bottom,self.top,self.name))

def add_cell(cellComplex,Cell):
    dim=Cell.dim
    cellComplex[dim].add(Cell)

def add_cell_and_cone(cellComplex,Cell):
    dim=Cell.dim
    cellComplex[dim].add(Cell)
    cellComplex[dim+1].add(ConeCell(Cell))
```

```python

def cells_of_tower(beta,mon):
    l=len(beta.braids)
    result={i:set({}) for i in range(5)}
    k = beta.k
    add_cell(result,TowerCell(2,"lambda",None,beta.sing_point,(None,None),mon))
    add_cell(result,TowerCell(2,"mu",None,beta.sing_point,(None,None),mon))
    add_cell_and_cone(result,TowerCell(3,"Hsup",None,beta.sing_point,(None,None),mon))
    for i in range(1,k+1):
        add_cell(result,TowerCell(0,"A",0,beta.sing_point,(i,None),mon))
        add_cell(result,TowerCell(0,"A",beta.strands+1,beta.sing_point,(i,None),mon))
        add_cell(result,TowerCell(1,"e",0,beta.sing_point,(i,None),mon))
        add_cell(result,TowerCell(1,"e",beta.strands+1,beta.sing_point,(i,None),mon))
        add_cell_and_cone(result,TowerCell(1,"m2",None,beta.sing_point,(i,None),mon))
        add_cell_and_cone(result,TowerCell(2,"varkappa",None,beta.sing_point,(i,None),mon))
    for r in range(1,l+1):
        add_cell(result,ConeCell(TowerCell(2,"lambda",None,beta.sing_point,(None,r),mon)))
        add_cell(result,ConeCell(TowerCell(2,"mu",None,beta.sing_point,(None,r),mon)))
        add_cell(result,TowerCell(0,"AA",None,beta.sing_point,(None,r),mon))
        add_cell_and_cone(result,TowerCell(3,"Hinf",None,beta.sing_point,(None,r),mon))
        for i in range(1,k+1):
            add_cell_and_cone(result,TowerCell(1,"m1",None,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(1,"d",0,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(2,"varsigma",0,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(2,"theta",None,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(2,"vartheta",None,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(2,"kappa",None,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(3,"PI",None,beta.sing_point,(i,r),mon))
            add_cell_and_cone(result,TowerCell(3,"OMEGA",None,beta.sing_point,(i,r),mon))
            add_cell(result,ConeCell(TowerCell(0,"A",0,beta.sing_point,(i,r),mon)))
            add_cell(result,ConeCell(TowerCell(0,"A",beta.strands+1,beta.sing_point,(i,r),mon)))
```

```
            add_cell(result,ConeCell(TowerCell(1,"e",0,beta.sing_point,(i,r),mon)))
            add_cell(result,ConeCell(TowerCell(1,"e",beta.strands+1,beta.sing_point,(i,r),mon)))
            for j in range(1,beta.n[r-1]+1):
                add_cell_and_cone(result,TowerCell(0,"A",j,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(1,"e",j,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(1,"d",j,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(2,"varsigma",j,beta.sing_point,(i,r),mon))
            if beta.braids[r-1][i-1]!=0:
                add_cell_and_cone(result,TowerCell(1,"hq",None,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(1,"hq1",None,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(2,"nu",1,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(2,"nu",2,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(2,"nu",3,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(2,"nu",4,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(3,"PHIpi",None,beta.sing_point,(i,r),mon))
                add_cell_and_cone(result,TowerCell(3,"PHIomega",None,beta.sing_point,(i,r),mon))
    for r in range(1,l):
        add_cell(result,ConeCell(TowerCell(3,"Hsup",None,beta.sing_point,(None,r),mon)))
        for i in range(1,k+1):
            add_cell(result,ConeCell(TowerCell(1,"m2",None,beta.sing_point,(i,r),mon)))
            add_cell(result,ConeCell(TowerCell(2,"varkappa",None,beta.sing_point,(i,r),mon)))
    return result

def top(Cell):
    #Cell is a BottomCell. returns its top
    c = TopCell(Cell.dim, Cell.name,Cell.r , Cell.jr, Cell.sing_point, Cell.i,Cell.mon)
    return c

def product(Cell):
    #Cell is a BottomCell. returns its product
    t = top(Cell)
    c = ProductCell(Cell,t)
    return c
```

```
def add_top_bottom_and_product(cellComplex,Cell):
     #Cell is a BottomCell (the Bottom). Adds Cell and its corresponding top and product
     prod = product(Cell)
     cellComplex[Cell.dim].add(Cell)
     cellComplex[Cell.dim+1].add(prod)
     cellComplex[Cell.dim].add(prod.top)

def cells_of_bridge(beta,mon):
    n=sum(beta.n)
    l=len(beta.braids)
    kc = beta.kc
    result={i:set({}) for i in range(5)}
    add_top_bottom_and_product(result, BottomCell(1,"m1",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(1,"m2",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(2,"theta",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(2,"vartheta",None,None,beta.sing_point, kc+1,mon))
    for j in range(0,n+1):
        add_top_bottom_and_product(result, BottomCell(1,"d",j,None,beta.sing_point, kc+1,mon))
    for j in range(0,n+2):
        add_top_bottom_and_product(result, BottomCell(0,"A",j,None,beta.sing_point, kc+1,mon))
    for i in range(1,kc+1):
          add_top_bottom_and_product(result, BottomCell(1,"m1",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(1,"m2",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(2,"theta",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(2,"vartheta",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(2,"kappa",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(2,"varkappa",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(3,"PI",None,None,beta.sing_point, i,mon))
          add_top_bottom_and_product(result, BottomCell(3,"OMEGA",None,None,beta.sing_point, i,mon))
          for j in range(0,n+1):
              add_top_bottom_and_product(result, BottomCell(1,"d",j,None,beta.sing_point, i,mon))
              add_top_bottom_and_product(result, BottomCell(2,"varsigma",j,None,beta.sing_point, i,mon))
          for j in range(0,n+2):
              add_top_bottom_and_product(result, BottomCell(0,"A",j,None,beta.sing_point, i,mon))
```

```
              add_top_bottom_and_product(result, BottomCell(1,"e",j,None,beta.sing_point, i,mon))
          if beta.conj_braid[i-1]!=0:
                add_top_bottom_and_product(result,BottomCell(1,"hq",None,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(1,"hq1",None,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(2,"nu",1,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(2,"nu",2,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(2,"nu",3,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(2,"nu",4,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(3,"PHIpi",None,None,beta.sing_point,i,mon))
                add_top_bottom_and_product(result,BottomCell(3,"PHIomega",None,None,beta.sing_point,i,mon))


    add_top_bottom_and_product(result, BottomCell(1,"w0",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(1,"w1",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(2,"psi",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(2,"xi",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(3,"PSI",None,None,beta.sing_point, kc+1,mon))
    add_top_bottom_and_product(result, BottomCell(3,"XI",None,None,beta.sing_point, kc+1,mon))
    for r in range(1,l+1):
        add_top_bottom_and_product(result, BottomCell(1,"z",r,beta.n[r-1],beta.sing_point, kc+1,mon))
        add_top_bottom_and_product(result, BottomCell(2,"phi",r,None,beta.sing_point, kc+1,mon))
        add_top_bottom_and_product(result, BottomCell(2,"omega",r,None,beta.sing_point, kc+1,mon))
        for j in range(1,beta.n[r-1]):
            add_top_bottom_and_product(result, BottomCell(1,"z",r,j,beta.sing_point, kc+1,mon))
            add_top_bottom_and_product(result, BottomCell(2,"zeta",r,j,beta.sing_point, kc+1,mon))
    for r in range(1,l):
        add_top_bottom_and_product(result, BottomCell(3,"LAMDA",r,None,beta.sing_point, kc+1,mon))

    add_top_bottom_and_product(result, BottomCell(1,"w0",None,None,beta.sing_point, 0,mon))
```

```
    add_top_bottom_and_product(result, BottomCell(1,"w1",None,None,beta.sing_point, 0,mon))
    add_top_bottom_and_product(result, BottomCell(2,"psi",None,None,beta.sing_point, 0,mon))
    add_top_bottom_and_product(result, BottomCell(2,"xi",None,None,beta.sing_point, 0,mon))
    add_top_bottom_and_product(result, BottomCell(3,"PSI",None,None,beta.sing_point, 0,mon))
    add_top_bottom_and_product(result, BottomCell(3,"XI",None,None,beta.sing_point, 0,mon))

    for r in range(1,n+1):
        add_top_bottom_and_product(result, BottomCell(1,"z",r,1,beta.sing_point, 0,mon))
        add_top_bottom_and_product(result, BottomCell(2,"phi",r,None,beta.sing_point, 0,mon))
        add_top_bottom_and_product(result, BottomCell(2,"omega",r,None,beta.sing_point, 0,mon))
    for r in range(1,n):
        add_top_bottom_and_product(result, BottomCell(3,"LAMDA",r,None,beta.sing_point, 0,mon))

    return result

def join_cells(l):
    # l is a list of cellular complex as is given by cells_of_tower and cells_of_bridge
    result={}
    for i in range(5):
        result[i] = reduce(lambda x, y : x |y, [d[i] for d in l])
    return result

def cannonize(c,cwComp):
    #c is a set of CellWithSign
    for a_Cell in c:
        exist_Cell = get_equivalent(cwComp[a_Cell.Cell.dim],a_Cell.Cell)
        a_Cell.Cell = exist_Cell

def euler(cwComplex):
    i=1
    result = 0
    for dim in cwComplex:
        result += i*len(cwComplex[dim])
        i *= -1
    return result

def in_curve(c):
    if isinstance(c,ProductCell):
        return  in_curve(c.top)
    elif isinstance(c,TowerCell):
        if c.name == "A":
            beta = c.mon.all_braids[c.sing_point]
```

```
            n=sum(beta.n)
            return not (c.index == 0 or c.index ==n+1)
        if c.name == "e":
            beta = c.mon.all_braids[c.sing_point]
            n=sum(beta.n)
            if c.index == 0 or c.index == n+1:
                return False
            else:
                q = abs(beta.braids[c.r-1][c.i-1])
                return c.index!=q and c.index!=q+1
    elif isinstance(c,BottomCell) or isinstance(c,TopCell):
        if c.name == "A":
            beta = c.mon.all_braids[c.sing_point]
            n=sum(beta.n)
            return not (c.r == 0 or c.r ==n+1)
        if c.name == "e":
            beta = c.mon.all_braids[c.sing_point]
            n=sum(beta.n)
            q = abs(beta.conj_braid[c.i-1])
            return not (c.r == 0 or c.r ==n+1 or c.r ==q or c.r == q+1)

        elif c.name in ["hq","hq1","z","AA"]:
            return True
    elif isinstance(c,ConeCell):
        return in_curve(c.base) and (not c.name == "AA")
    return False

def error_test( cwComplex1):
    for dim in cwComplex1:
       print len(cwComplex1[dim])
    cells_without_border = []
    cells_with_error_in_border = []
    cells_in_a_border_but_not_in_complex = []
    cells_given_a_cell_in_a_border_but_not_in_complex = {}

    for dim in cwComplex1:
        for c in cwComplex1[dim]:
            try:
              borde = c.border()
              if borde == None:
                cells_without_border.append(c)
              else:
                for b in borde:
                    if b.Cell not in cwComplex1[b.Cell.dim]:
                        #print "Cell:"+str(c.__dict__)+"  have Cell:"+str(b.Cell.__dict__)+"in his border that isn't in the complex"
                        cells_in_a_border_but_not_in_complex.append(b.Cell)
                        if c in cells_given_a_cell_in_a_border_but_not_in_complex:
                            cells_given_a_cell_in_a_border_but_not_in_complex[c].add(b)
                        else:
```

```
    cells_given_a_cell_in_a_border_but_not_in_complex[c]={b}
            except Exception as e:
              cells_with_error_in_border.append(c)
    print euler(cwComplex1)
    return (cells_with_error_in_border,
            cells_without_border,
            cells_in_a_border_but_not_in_complex,
            cells_given_a_cell_in_a_border_but_not_in_complex)
```

*Appendix B*

# *Code for the Simplicial Decomposition for Affine Plane Curves*

Here we exhibit the code of the program in SageMath that turns the CW decomposition of $(\mathcal{D}, \Omega \cap \mathcal{D})$ into a simplicial decomposition. This program was explained in Section 2.2.

```

class Simple_Cell(object):
    def __init__(self,dim,name):
        self.dim = dim
        self.name = name
        self.borde = set({})

    def __str__(self):
        return self.name

    def __repr__(self):
        return self.name

    def border(self):
        return self.borde

    def set_border(self,borde):
        self.borde = borde

class Cell_With_Sign(object):
    def __init__(self,cell,sgn):
```

```
        self.cell = cell
        self.sgn =sgn
    def __repr__(self):
        if self.sgn==1:
           return self.cell.__repr__()
        else:
            return "-"+self.cell.__repr__()
    def __eq__(self,other):
        return isinstance(other,Cell_With_Sign) and (self.sgn==other.sgn)
    and (self.cell==self.cell)
    def __hash__(self):
        return hash((self.cell,self.sgn))
    def cone(self):
        return Cell_With_Sign(ConeCell(self.cell),self.sgn)
    def product(self):
       if isinstance(self.cell,BottomCell):
          return Cell_With_Sign(product(self.cell),self.sgn)
       else:
           raise Exception("A ProductCell must have a BottomCell as a base"
    )

def simple_complex(cwComplex):
    simple_dict = {}
    result = {dim:set({}) for dim in cwComplex}
    for dim in cwComplex:
        for c in cwComplex[dim]:
            new_Simple_Cell = Simple_Cell(dim,c.name)
            new_Simple_Cell.from_monod = c
            simple_dict[c] = new_Simple_Cell
            result[dim].add(new_Simple_Cell)
    for dim in result:
        for simp_cell in result[dim]:
            b = simp_cell.from_monod.border()
            for e in b:
                e.cell = simple_dict[e.cell]
            simp_cell.set_border(b)
    return result

def subcomplex(c):
    return in_curve(c.from_monod)

def from_CW_to_simplicial(cell_complex,subcomplex):
    def in_X(c):
        '''

        '''
        if c.dim==1:
            result = [subcomplex(e.cell) for e in c.border()] == [True,True
    ]
        else:
            result = False
        return result
```

```
    def make_vs_and_ws(cell_complex):
        v={}
        w={}
        B={}
        dim=max([d for d in cell_complex])
        X=set({c for c in cell_complex[1] if in_X(c) })
        for c in cell_complex[0]:
            v[(0,c)]=set({})
            w[(0,c)]={(c,)}
            B[(0,c)]={(c,)}
        for c in cell_complex[1]:
            borde = c.border()
            B[(0,c)]={(b.cell,) for b in borde}
            if c in X:
                v[(0,c)]={(c,)}
                v[(1,c)]={(c,)+l for l in B[(0,c)]}
            else:
                v[(0,c)]=set({})
                v[(1,c)]={tuple((b.cell for b in borde))}
            w[(0,c)]=v[(0,c)].union(B[(0,c)])
            w[(1,c)]=v[(1,c)]
        for d in range(2,dim+1):
            for c in cell_complex[d]:
                v[(0,c)]={(c,)}
                B[(0,c)]=set({})
                borde = c.border()
                if borde == None:
                    print "Empty border"
                for b in borde:
                    if b.cell not in cell_complex[d-1]:
                        print "Border cell not in complex"
                    global recuperaCelda
                    recuperaCelda=b
                    B[(0,c)].update(w[(0,b.cell)])
                w[(0,c)]=v[(0,c)].union(B[(0,c)])
                for i in range(1,d):
                    v[(i,c)]={(c,)+l for l in B[(i-1,c)]}
                    B[(i,c)]=set({})
                    for b in borde:
                        B[(i,c)].update(w[(i,b.cell)])
                    w[(i,c)]=v[(i,c)].union(B[(i,c)])
                v[(d,c)]={(c,)+l for l in B[(d-1,c)]}
                w[(d,c)]=set(v[(d,c)])
        return v,w,B
    v,w,B=make_vs_and_ws(cell_complex)
    sim_comp = {0:set({}),1:set({}),2:set({}),3:set({}),4:set({})}
    for dim in cell_complex:
        for c in cell_complex[dim]:
            for dim1 in xrange(0,dim+1):
                for simplex in v[(dim1,c)]:
                     sim_comp[dim1].add(simplex)
    for c in cell_complex[0]:
```

```
        sim_comp[0].add((c,))
    return sim_comp

def from_CW_to_simplicial_with_sets(cell_complex,subcomplex):
    def in_X(c):
        '''

        '''
        if c.dim==1:
            result = [subcomplex(e.cell) for e in c.border()] == [True,True] and not subcomplex(c)
        else:
            result = False
        return result
    def make_vs_and_ws(cell_complex):
        v={}
        w={}
        B={}
        dim=max([d for d in cell_complex])
        X=set({c for c in cell_complex[1] if in_X(c) })
        for c in cell_complex[0]:
            v[(0,c)]=set({})
            w[(0,c)]={frozenset({c})}
            B[(0,c)]={frozenset({c})}
        for c in cell_complex[1]:
            borde = c.border()
            B[(0,c)]={frozenset({b.cell}) for b in borde}
            if c in X:
                v[(0,c)]={frozenset({c})}
                v[(1,c)]={l.union({c}) for l in B[(0,c)]}
            else:
                v[(0,c)]=set({})
                v[(1,c)]={frozenset([b.cell for b in borde])}
            w[(0,c)]=v[(0,c)].union(B[(0,c)])
            w[(1,c)]=v[(1,c)]
        for d in range(2,dim+1):
            for c in cell_complex[d]:
                v[(0,c)]={frozenset({c})}
                B[(0,c)]=set({})
                borde = c.border()
                if borde == None:
                    print "Empty border"
                for b in borde:
                    if b.cell not in cell_complex[d-1]:
                        print "Border cell not in complex"
                    global recuperaCelda
                    recuperaCelda=b
                    B[(0,c)].update(w[(0,b.cell)])
                w[(0,c)]=v[(0,c)].union(B[(0,c)])
                for i in range(1,d):
                    v[(i,c)]={l.union({c}) for l in B[(i-1,c)]}
                    B[(i,c)]=set({})
```

```
                    for b in borde:
                        B[(i,c)].update(w[(i,b.cell)])
                    w[(i,c)]=v[(i,c)].union(B[(i,c)])
                v[(d,c)]={l.union({c}) for l in B[(d-1,c)]}
                w[(d,c)]=set(v[(d,c)])
        return v,w,B
    v,w,B=make_vs_and_ws(cell_complex)
    sim_comp = {0:set({}),1:set({}),2:set({}),3:set({}),4:set({})}
    for dim in cell_complex:
        for c in cell_complex[dim]:
            for dim1 in xrange(0,dim+1):
                for simplex in v[(dim1,c)]:
                     sim_comp[dim1].add(simplex)
    for c in cell_complex[0]:
        sim_comp[0].add(frozenset({c}))
    return sim_comp

def simplex_in_subcomplex(simplex, subcomplex):
        for c in simplex:
            if not subcomplex(c):
                return False
        return True

def write_elem(e):
    if isinstance(e,frozenset):
       write_simp(e)
    else:
       print e,

def write_simp(simp):
    print "(",
    for e in simp:
        write_elem(e)
        print ",",
    print ")",

def write_simplex_set(s, dim):
    print ("dim={}:"+(10*"-")).format(dim)
    for simp in s:
        write_simp(simp)
    print

def write_simp_complex(complex):
    for dim in complex:
        write_simplex_set(complex[dim],dim)

def border(simplex):
        result = set({})
        for c in simplex:
            result.add(simplex.difference({c}))
        return result
```

```
def subdivide(cell_complex):

    def make_vs_and_ws(cell_complex):
        v={}
        w={}
        B={}
        dim=max([d for d in cell_complex])
        dims=sorted([d for d in cell_complex])
        for c in cell_complex[0]:
            v[(0,c)]=set({})
            w[(0,c)]={frozenset({c})}
            B[(0,c)]={frozenset({c})}
        for d in range(1,dim+1):
            for c in cell_complex[d]:
                v[(0,c)]={frozenset({c})}
                w[(0,c)]=set({})
                B[(0,c)]=set({})
                borde = border(c)
                if borde == None:
                    print type(c),c.__dict__,c.bottom.name
                for b in borde:
                    global recuperaCelda
                    recuperaCelda=b
                    B[(0,c)].update(w[(0,b)])
                w[(0,c)]=v[(0,c)].union(B[(0,c)])
                for i in range(1,d):
                    v[(i,c)]={l.union({c}) for l in B[(i-1,c)]}
                    B[(i,c)]=set({})
                    for b in borde:
                        B[(i,c)].update(w[(i,b)])
                        w[(i,c)]=v[(i,c)].union(B[(i,c)])
                v[(d,c)]={l.union({c}) for l in B[(d-1,c)]}
                w[(d,c)]=set(v[(d,c)])
        return v,w,B
    v,w,B=make_vs_and_ws(cell_complex)
    sim_comp = {0:set({}),1:set({}),2:set({}),3:set({}),4:set({})}
    for dim in cell_complex:
        for c in cell_complex[dim]:
            for dim1 in xrange(0,dim+1):
                for simplex in v[(dim1,c)]:
                     sim_comp[dim1].add(simplex)
    for c in cell_complex[0]:
        sim_comp[0].add(frozenset({c}))
    return sim_comp

def subsimplex_in_subcomplex(subsimplex, subcomplex):
        for c in subsimplex:
            if not simplex_in_subcomplex(c, subcomplex):
                return False
        return True

def subsimplex_in_reg_neig(subsimplex, subcomplex):
```

```
        for c in subsimplex:
            if simplex_in_subcomplex(c, subcomplex):
                return True
        return False

def comp_reg_neig(sim_comp,subcomplex):
    return {i:{c for c in sim_comp[i] if not subsimplex_in_reg_neig(c,
   subcomplex)} for i in sim_comp}

def reg_neig(sim_comp,subcomplex):
    return {i:{c for c in sim_comp[i] if  subsimplex_in_reg_neig(c,
   subcomplex)} for i in sim_comp}
```

*Appendix C*

# *Code for the Calculation of the Complex Homology*

Here we exhibit the code of the program in SageMath used to calculate the ranks of the matrices $[\partial^{(r)}_{i,j,k}]_R(\mathbf{t},\mathbf{s},\mathbf{u})$ through Lemmas 4.17 to 4.24.

```
#!/usr/bin/env python
# coding: utf-8

# In[ ]:




# $$
# \def\CC{\bf C}
# \def\QQ{\bf Q}
# \def\RR{\bf R}
# \def\ZZ{\bf Z}
# \def\NN{\bf N}
# $$
#
# The ring $AN$ is the base ring over which we are going to work.
# In fact, the ring that interest us is the ring
# $R_{a,b,m}=\mathbb{C}[t,s,u]/(t^a-1,s^b-1,u^m-1)$, but we work with
# $AN$ for practical reasons. We define another rings that we will use
# too, and lists of keys that will be used to store the data orderly.

# In[ ]:


```

```
ANs.<s>=QQ[]
ANt.<t>=QQ[]
ANu.<u>=QQ[]
ANtu.<t,u>=QQ[]
ANsu.<s,u>=QQ[]
ANst.<s,t>=QQ[]
AN.<t,s,u>=QQ[]
claves=[(s,t,u), (t,u),  (s,u), (s,t),(u,)]
anillos={(s,t,u):AN, (t,u):ANtu,  (s,u):ANsu, (s,t):ANst,(u,):ANu, (s,):ANs, (t,):ANt}

# In[ ]:


M=[FreeModule(AN,_) for _ in [8,23,25,12,3]]

# We start with the module $M_0$ of the $0$-cells; although we define it
# as a free module for practical reasons, it is not actually free.
# We name the elements of the generating system and distribute them according
# to the actions that act trivially upon them.
#
# The module $M_0$ is generated by
# $R,P_1,\hat{P}_1,P_2,\hat{P}_2,Q_1,Q_2,\hat{R}$. In fact,
#
# $$M_0=R_{a,b,m}\langle R\rangle\oplus R_{a,b,m}/(s-1)\langle P_1,\hat{P}_1\rangle\oplus R_{a,b,m}/(t-1)\langle P_2,\hat{P}_2\rangle\oplus R_{a,b,m}/(u-1)\langle Q_1,Q_2,\hat{R}\rangle$$
#
# The dictionary **M0dic** assigns to each key the elements of the generating
# system upon which the variables appearing in the key do **NOT** act trivially.

# In[ ]:


M0=M[0]
R,P1,P1h,P2,P2h,Q1,Q2,Rh=M0.gens()
Mdic={(0,s,t,u):[R],(0,t,u):[P1,P1h],(0,s,u):[P2,P2h],(0,s,t):[Q1,Q2,Rh],(0,u):[]}
Ldic={}
i=0
for _ in claves:
    cl=tuple([0]+list(_))
    j=len(Mdic[cl])
    Ldic[cl]=range(i,i+j)
    i=i+j

# We do the same for the module $M_1$ :
#
# The module $M_1$ is generated by
```

```
# $m,l,k,a_1,a_2,b_1,b_2,\hat{m},\hat{l},\hat{a}_1,\hat{a}_2,r,c_1,\hat{c}_1,p_1,c_2,\hat{c}_2,p_2,h_1,h_2,\hat{k},q_1,q_2$.
# In fact,
#
# $$M_1=R_{a,b,m}\langle m,l,k,a_1,a_2,b_1,b_2,\hat{m},\hat{l},\hat{a}_1,\hat{a}_2,r\rangle\oplus R_{a,b,m}/(s-1)\langle c_1,\hat{c}_1,p_1\rangle\oplus R_{a,b,m}/(t-1)\langle c_2,\hat{c}_2,p_2\rangle\oplus R_{a,b,m}/(u-1)\langle h_1,h_2,\hat{k},q_1,q_2\rangle$$

# In[ ]:


M1=M[1]
m,l,k,a1,a2,b1,b2,mh,lh,a1h,a2h,r,c1,c1h,p1,c2,c2h,p2,h1,h2,kh,q1,q2=M1.gens()
Mdic.update({(1,s,t,u):[m,l,k,a1,a2,b1,b2,mh,lh,a1h,a2h,r], (1,t,u):[c1,c1h,p1], (1,s,u):[c2,c2h,p2], (1,s,t):[h1,h2,kh,q1,q2], (1,u):[]})
i=0
for _ in claves:
    cl=tuple([1]+list(_))
    j=len(Mdic[cl])
    Ldic[cl]=range(i,i+j)
    i=i+j

# Now we build the matrix for $\delta_1:M_1\to M_0$. We do it in such
# a way that it will be easy to recover later the matrices that we will
# need according to the isotropy of the generators.

# In[ ]:


imagenes={}
pr=(s,t,u)
mm=len(Mdic[tuple([1]+list(pr))])
imagenes[1,0,pr,(s,t,u)]=[(u*s-1)*R,(u*t-1)*R,(t-s)*R,M0(0),M0(0),-R,-R,M0(0),M0(0),M0(0),M0(0),R]
imagenes[1,0,pr,(t,u)]=[M0(0),M0(0),M0(0),P1,M0(0),M0(0),M0(0),M0(0),M0(0),P1h,M0(0),M0(0)]
imagenes[1,0,pr,(s,u)]=[M0(0),M0(0),M0(0),M0(0),P2,M0(0),M0(0),M0(0),M0(0),M0(0),P2h,M0(0)]
imagenes[1,0,pr,(s,t)]=[M0(0),M0(0),M0(0),-Q1,-Q2,Q1,Q2,(s-1)*Rh,(t-1)*Rh,-Rh,-Rh,-Rh]
imagenes[1,0,pr,(u,)]=mm*[M0(0)]
imagenes[1,0,pr]=[sum([_[i] for _ in [imagenes[1,0,pr,cl] for cl in claves]]) for i in range(mm)]
pr=(t,u)
mm=len(Mdic[tuple([1]+list(pr))])
imagenes[1,0,pr,(s,t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(t,u)]=[(t-1)*P1,(t-1)*P1h,P1-P1h]
imagenes[1,0,pr,(s,u)]=mm*[M0(0)]
imagenes[1,0,pr,(s,t)]=mm*[M0(0)]
imagenes[1,0,pr,(u,)]=mm*[M0(0)]
```

```
imagenes[1,0,pr]=[sum([_[i] for _ in [imagenes[1,0,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(s,u)
mm=len(Mdic[tuple([1]+list(pr))])
imagenes[1,0,pr,(s,t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(s,u)]=[(s-1)*P2,(s-1)*P2h,P2-P2h]
imagenes[1,0,pr,(s,t)]=mm*[M0(0)]
imagenes[1,0,pr,(u,)]=mm*[M0(0)]
imagenes[1,0,pr]=[sum([_[i] for _ in [imagenes[1,0,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(s,t)
mm=len(Mdic[tuple([1]+list(pr))])
imagenes[1,0,pr,(s,t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(s,u)]=mm*[M0(0)]
imagenes[1,0,pr,(s,t)]=[(t-s)*Q1,-(t-s)*Q2,(t-s)*Rh,Q1-Rh,Q2-Rh]
imagenes[1,0,pr,(u,)]=mm*[M0(0)]
imagenes[1,0,pr]=[sum([_[i] for _ in [imagenes[1,0,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(u,)
mm=len(Mdic[tuple([1]+list(pr))])
imagenes[1,0,pr,(s,t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(t,u)]=mm*[M0(0)]
imagenes[1,0,pr,(s,u)]=mm*[M0(0)]
imagenes[1,0,pr,(s,t)]=mm*[M0(0)]
imagenes[1,0,pr,(u,)]=mm*[M0(0)]
imagenes[1,0,pr]=[sum([_[i] for _ in [imagenes[1,0,pr,cl] for cl in claves
    ]]) for i in range(mm)]
imagenes[1,0]=flatten([imagenes[1,0,_] for _ in claves])
delta1=M1.hom(imagenes[1,0],M0)
A={}
A[1]=delta1.matrix()
A1=A[1]

# In[ ]:


show(A1)

# In[ ]:


latex(A1)

# In[ ]:


def varclave(tuplevar,clave):
    res=True
    for _ in tuplevar:
        res=res and _ in clave
```

```
    return res
tupleclaves=[(s,),(t,),(u,),(s,t),(t,u),(s,u),(s,t,u)]
clavevar={tuplevar:[_ for _ in claves if varclave(tuplevar,_)] for tuplevar in tupleclaves}

# In[ ]:


for pr in tupleclaves:
    clpr=clavevar[pr]
    sbs={vr:1 for vr in (s,t,u) if vr not in pr}
    pr0=flatten([Ldic[tuple([0]+list(_))] for _ in clpr])
    A[1,pr]=Matrix(flatten([imagenes[1,0,_] for _ in clpr])).matrix_from_columns(pr0).subs(sbs).change_ring(anillos[pr])

# We do the same for $M_2$ :
#
# The module $M_2$ is generated by
# $\sigma, \pi, \theta_1, \theta_2, \omega_1, \omega_2, \phi_1, \phi_2, \hat{\sigma}, \hat{\pi}, \hat{\theta}_1, \hat{\theta}_2, \hat{\omega}_1, \hat{\omega}_2, \mu, \lambda, \kappa, \alpha_1, \alpha_2, \beta_1, \beta_2, \zeta_1, \zeta_2, \eta_1, \eta_2$.
# In fact,
#
# $$M_2=R_{a,b,m}\langle \sigma, \pi, \theta_1, \theta_2, \omega_1, \omega_2, \phi_1, \phi_2, \hat{\sigma}, \hat{\pi}, \hat{\theta}_1, \hat{\theta}_2, \hat{\omega}_1, \hat{\omega}_2, \mu, \lambda, \kappa, \alpha_1, \alpha_2, \beta_1, \beta_2\rangle\oplus R_{a,b,m}/(s-1)\langle \zeta_1\rangle\oplus R_{a,b,m}/(t-1)\langle \zeta_2\rangle\oplus R_{a,b,m}/(u-1)\langle \eta_1, \eta_2\rangle$$

# In[ ]:


M2=M[2]
sigma,pi_0,theta_1,theta_2,omega_1,omega_2,phi_1,phi_2,sigmah,pi_0h,theta_1h,theta_2h,omega_1h,omega_2h,mu,lambda_0,kappa,alpha_1,alpha_2,beta_1,beta_2,zeta_1,zeta_2,eta_1,eta_2=M2.gens()
Mdic.update({(2,s,t,u):[sigma,pi_0,theta_1,theta_2,omega_1,omega_2,phi_1,phi_2, sigmah,pi_0h,theta_1h,theta_2h,omega_1h,omega_2h,mu,lambda_0,kappa,alpha_1,alpha_2,beta_1,beta_2], (2,t,u):[zeta_1], (2,s,u):[zeta_2], (2,s,t):[eta_1,eta_2], (2,u):[]})
i=0
for _ in claves:
    cl=tuple([2]+list(_))
    j=len(Mdic[cl])
    Ldic[cl]=range(i,i+j)
    i=i+j

# In[ ]:


```

```
pr=(s,t,u)
mm=len(Mdic[tuple([2]+list(pr))])
imagenes[2,1,pr,(s,t,u)]=[s*l-t*m-k, m+u*k-l, m+(s-1)*a1+(s*u-1)*b1, l+(t-1)*a2+(t*u-1)*b2,-l+(1-t)*a1+(1-t*u)*b1, -m+(1-s)*a2+(1-s*u)*b2, -k+(s-t)*b1, k-(s-t)*b2, s*lh-t*mh, mh-lh, mh+(s-1)*a1h, lh+(t-1)*a2h, -lh+(1-t)*a1h, -mh+(1-s)*a2h, (u*s-1)*r+mh-m, (u*t-1)*r+lh-l, (t-s)*r-k, a1h-a1, a2h-a2, -r-b1, -r-b2]
imagenes[2,1,pr,(t,u)]=[M1(0),M1(0),M1(0),M1(0),c1,M1(0),M1(0),M1(0),M1(0),M1(0),M1(0),M1(0),c1h,M1(0),M1(0),M1(0),M1(0),p1,M1(0),M1(0),M1(0)]
imagenes[2,1,pr,(s,u)]=[M1(0),M1(0),M1(0),M1(0),M1(0),c2,M1(0),M1(0),M1(0),M1(0),M1(0),M1(0),M1(0),c2h,M1(0),M1(0),M1(0),M1(0),p2,M1(0),M1(0)]
imagenes[2,1,pr,(s,t)]=[M1(0),M1(0),M1(0),M1(0),M1(0),M1(0),h1,h2,-kh,u*kh,M1(0),M1(0),M1(0),M1(0),M1(0),M1(0),kh,-q1,-q2,q1,q2]
imagenes[2,1,pr,(u,)]=mm*[M1(0)]
imagenes[2,1,pr]=[sum([_[i] for _ in [imagenes[2,1,pr,cl] for cl in claves]]) for i in range(mm)]
pr=(t,u)
mm=len(Mdic[tuple([2]+list(pr))])
imagenes[2,1,pr,(s,t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(t,u)]=[(t-1)*p1+c1h-c1]
imagenes[2,1,pr,(s,u)]=mm*[M1(0)]
imagenes[2,1,pr,(s,t)]=mm*[M1(0)]
imagenes[2,1,pr,(u,)]=mm*[M1(0)]
imagenes[2,1,pr]=[sum([_[i] for _ in [imagenes[2,1,pr,cl] for cl in claves]]) for i in range(mm)]
pr=(s,u)
mm=len(Mdic[tuple([2]+list(pr))])
imagenes[2,1,pr,(s,t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(s,u)]=[(s-1)*p2+c2h-c2]
imagenes[2,1,pr,(s,t)]=mm*[M1(0)]
imagenes[2,1,pr,(u,)]=mm*[M1(0)]
imagenes[2,1,pr]=[sum([_[i] for _ in [imagenes[2,1,pr,cl] for cl in claves]]) for i in range(mm)]
pr=(s,t)
mm=len(Mdic[tuple([2]+list(pr))])
imagenes[2,1,pr,(s,t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(s,u)]=mm*[M1(0)]
imagenes[2,1,pr,(s,t)]=[(t-s)*q1+kh-h1,-(t-s)*q2-kh-h2]
imagenes[2,1,pr,(u,)]=mm*[M1(0)]
imagenes[2,1,pr]=[sum([_[i] for _ in [imagenes[2,1,pr,cl] for cl in claves]]) for i in range(mm)]
pr=(u,)
mm=len(Mdic[tuple([2]+list(pr))])
imagenes[2,1,pr,(s,t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(t,u)]=mm*[M1(0)]
imagenes[2,1,pr,(s,u)]=mm*[M1(0)]
imagenes[2,1,pr,(s,t)]=mm*[M1(0)]
imagenes[2,1,pr,(u,)]=mm*[M1(0)]
imagenes[2,1,pr]=[sum([_[i] for _ in [imagenes[2,1,pr,cl] for cl in claves]]) for i in range(mm)]
```

```
imagenes[2,1]=flatten([imagenes[2,1,_] for _ in claves])
delta2=M2.hom(imagenes[2,1],M1)
A[2]=delta2.matrix()
A2=A[2]

# In[ ]:


show(A2)

# In[ ]:


latex(A2)

# In[ ]:


for pr in tupleclaves:
    clpr=clavevar[pr]
    sbs={vr:1 for vr in (s,t,u) if vr not in pr}
    pr0=flatten([Ldic[tuple([1]+list(_))] for _ in clpr])
    A[2,pr]=Matrix(flatten([imagenes[2,1,_] for _ in clpr])).
   matrix_from_columns(pr0).subs(sbs).change_ring(anillos[pr])

# The module $M_3$ is generated by
# $\Psi_1,\Psi_2,\hat{\Psi}_1,\hat{\Psi}_2,\Theta_1,\Theta_2,\Omega_1,\
    Omega_2,\Phi_1,\Phi_2,\Sigma,\Pi$
# and is free.

# In[ ]:


M3=M[3]
Psi_1,Psi_2,Psi_1h,Psi_2h,Theta_1,Theta_2,Omega_1,Omega_2,Phi_1,Phi_2,
    Sigma_0,Pi_0=M3.gens()
Mdic.update({(3,s,t,u):[Psi_1,Psi_2,Psi_1h,Psi_2h,Theta_1,Theta_2,Omega_1,
    Omega_2,Phi_1,Phi_2,Sigma_0,Pi_0], (3,t,u):[], (3,s,u):[], (3,s,t):[],
    (3,u):[]})
i=0
for _ in claves:
    cl=tuple([3]+list(_))
    j=len(Mdic[cl])
    Ldic[cl]=range(i,i+j)
    i=i+j

# In[ ]:


pr=(s,t,u)
mm=len(Mdic[tuple([3]+list(pr))])
```

```
imagenes[3,2,pr,(s,t,u)]=[sigma+pi_0+(t-1)*theta_1+(s-1)*omega_1+(u-1)*
    phi_1,-sigma-pi_0+(s-1)*theta_2+(t-1)*omega_2+(u-1)*phi_2,sigmah+pi_0h
    +(t-1)*theta_1h+(s-1)*omega_1h,-sigmah-pi_0h+(s-1)*theta_2h+(t-1)*
    omega_2h,mu+(s-1)*alpha_1+(s*u-1)*beta_1+theta_1-theta_1h,lambda_0+(t
    -1)*alpha_2+(t*u-1)*beta_2+theta_2-theta_2h,-lambda_0+(1-t)*alpha_1+(1-
    t*u)*beta_1+omega_1-omega_1h,-mu+(1-s)*alpha_2+(1-s*u)*beta_2+omega_2-
    omega_2h,-kappa+(s-t)*beta_1+phi_1, kappa-(s-t)*beta_2+phi_2,s*lambda_0
    -t*mu-kappa+sigma-sigmah,mu+u*kappa-lambda_0+pi_0-pi_0h]
imagenes[3,2,pr,(t,u)]=[M2(0),M2(0),M2(0),M2(0),M2(0),M2(0),zeta_1,M2(0),M2
    (0),M2(0),M2(0),M2(0)]
imagenes[3,2,pr,(s,u)]=[M2(0),M2(0),M2(0),M2(0),M2(0),M2(0),M2(0),zeta_2,M2
    (0),M2(0),M2(0),M2(0)]
imagenes[3,2,pr,(s,t)]=[M2(0),M2(0),M2(0),M2(0),M2(0),M2(0),M2(0),M2(0),
    eta_1,eta_2,M2(0),M2(0)]
imagenes[3,2,pr,(u,)]=mm*[M2(0)]
imagenes[3,2,pr]=[sum([_[i] for _ in [imagenes[3,2,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(t,u)
mm=len(Mdic[tuple([3]+list(pr))])
imagenes[3,2,pr,(s,t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,t)]=mm*[M2(0)]
imagenes[3,2,pr,(u,)]=mm*[M2(0)]
imagenes[3,2,pr]=[sum([_[i] for _ in [imagenes[3,2,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(s,u)
mm=len(Mdic[tuple([3]+list(pr))])
imagenes[3,2,pr,(s,t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,t)]=mm*[M2(0)]
imagenes[3,2,pr,(u,)]=mm*[M2(0)]
imagenes[3,2,pr]=[sum([_[i] for _ in [imagenes[3,2,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(s,t)
mm=len(Mdic[tuple([3]+list(pr))])
imagenes[3,2,pr,(s,t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,t)]=mm*[M2(0)]
imagenes[3,2,pr,(u,)]=mm*[M2(0)]
imagenes[3,2,pr]=[sum([_[i] for _ in [imagenes[3,2,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(u,)
mm=len(Mdic[tuple([3]+list(pr))])
imagenes[3,2,pr,(s,t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(t,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,u)]=mm*[M2(0)]
imagenes[3,2,pr,(s,t)]=mm*[M2(0)]
imagenes[3,2,pr,(u,)]=mm*[M2(0)]
```

```
imagenes[3,2,pr]=[sum([_[i] for _ in [imagenes[3,2,pr,cl] for cl in claves
    ]]) for i in range(mm)]
imagenes[3,2]=flatten([imagenes[3,2,_] for _ in claves])
delta3=M3.hom(imagenes[3,2],M2)
A[3]=delta3.matrix()
A3=A[3]

# In[ ]:


show(A3)

# In[ ]:


latex(A3)

# In[ ]:


for pr in tupleclaves:
    clpr=clavevar[pr]
    sbs={vr:1 for vr in (s,t,u) if vr not in pr}
    pr0=flatten([Ldic[tuple([2]+list(_))] for _ in clpr])
    A[3,pr]=Matrix(flatten([imagenes[3,2,_] for _ in clpr])).
    matrix_from_columns(pr0).subs(sbs).change_ring(anillos[pr])

# The last module, $M_4$, is generated by $\Xi_1,\Xi_2,\Upsilon$ :
#
# $$M_4=R_{a,b,c}\langle\Xi_1,\Xi_2\rangle\oplus R_{a,b,c}/(t-1,s-1)\langle
    \Upsilon\rangle$$
#
# In this case we don't insert the differential correctly because it
    involves
# a polynomial on $a,b$. In fact,
#
# $$\partial_4(\Upsilon)=\frac{(t^a-1)(s^b-1)}{(t-1)(s-1)}\left(\hat{\Psi}
    _1+\hat{\Psi}_2\right)$$

# In[ ]:


M4=M[4]
Xi_1,Xi_2,Upsilon=M4.gens()
Mdic.update({(4,s,t,u):[Xi_1,Xi_2], (4,t,u):[], (4,s,u):[], (4,s,t):[], (4,
    u):[Upsilon]})
i=0
for _ in claves:
    cl=tuple([4]+list(_))
    j=len(Mdic[cl])
    Ldic[cl]=range(i,i+j)
    i=i+j
```

```


# In[ ]:


pr=(s,t,u)
mm=len(Mdic[tuple([4]+list(pr))])
imagenes[4,3,pr,(s,t,u)]=[Sigma_0+Pi_0+(t-1)*Theta_1+(s-1)*Omega_1+(u-1)*
    Phi_1-Psi_1+Psi_1h,-Sigma_0-Pi_0+(s-1)*Theta_2+(t-1)*Omega_2+(u-1)*
    Phi_2-Psi_2+Psi_2h]
imagenes[4,3,pr,(t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,t)]=mm*[M3(0)]
imagenes[4,3,pr,(u,)]=mm*[M3(0)]
imagenes[4,3,pr]=[sum([_[i] for _ in [imagenes[4,3,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(t,u)
mm=len(Mdic[tuple([4]+list(pr))])
imagenes[4,3,pr,(s,t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,t)]=mm*[M3(0)]
imagenes[4,3,pr,(u,)]=mm*[M3(0)]
imagenes[4,3,pr]=[sum([_[i] for _ in [imagenes[4,3,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(s,u)
mm=len(Mdic[tuple([4]+list(pr))])
imagenes[4,3,pr,(s,t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,t)]=mm*[M3(0)]
imagenes[4,3,pr,(u,)]=mm*[M3(0)]
imagenes[4,3,pr]=[sum([_[i] for _ in [imagenes[4,3,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(s,t)
mm=len(Mdic[tuple([4]+list(pr))])
imagenes[4,3,pr,(s,t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,t)]=mm*[M3(0)]
imagenes[4,3,pr,(u,)]=mm*[M3(0)]
imagenes[4,3,pr]=[sum([_[i] for _ in [imagenes[4,3,pr,cl] for cl in claves
    ]]) for i in range(mm)]
pr=(u,)
mm=len(Mdic[tuple([4]+list(pr))])
imagenes[4,3,pr,(s,t,u)]=[Psi_1h+Psi_2h]
imagenes[4,3,pr,(t,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,u)]=mm*[M3(0)]
imagenes[4,3,pr,(s,t)]=mm*[M3(0)]
imagenes[4,3,pr,(u,)]=mm*[M3(0)]
imagenes[4,3,pr]=[sum([_[i] for _ in [imagenes[4,3,pr,cl] for cl in claves
    ]]) for i in range(mm)]
imagenes[4,3]=flatten([imagenes[4,3,_] for _ in claves])
```

```
delta4=M4.hom(imagenes[4,3],M3)
A[4]=delta4.matrix()
A4=A[4]

# In[ ]:


for pr in tupleclaves:
    clpr=clavevar[pr]
    sbs={vr:1 for vr in (s,t,u) if vr not in pr}
    pr0=flatten([Ldic[tuple([3]+list(_))] for _ in clpr])
    A[4,pr]=Matrix(flatten([imagenes[4,3,_] for _ in clpr])).
    matrix_from_columns(pr0).subs(sbs).change_ring(anillos[pr])

# In[ ]:


#P=AN((t^var_a-1)*(s^var_b-1)/(t-1)/(s-1))

# In[ ]:


for j in [1..4]:
    A[j,()]=A[j](t=1,s=1,u=1).change_ring(QQ)

# In the following cell we have the dimensions of the matrices of the
# differentials in the case $(\zeta,\xi,\mu)=(1,1,1)$.
#
# > $\dim C_0=8,\dots,\dim C_4=3$

# In[ ]:


for j in [1..4]:
    print A[j,()].dimensions()

# In the following cell we have the ranks of the matrices: 7,16,9,3

# In[ ]:


k=A[1,()].ncols()
for j in [1,2,3,4]:
    mat=A[j,()]
    #print "Rango de C",0,":",mat.ncols()
    print "Rango del ker de delta",j-1,":",k
    print "Rango de la imagen de delta",j,":",mat.rank()
    k=mat.nrows()-mat.rank()

# In[ ]:


```

```
[(mat.rank(),mat.ncols()) for mat in [A[j,()] for j in [4,3,2,1]]]

# The following cell examines the $(\zeta,\xi,\mu)=(\zeta,1,1)$, with
# $\zeta\neq 1$.
#
# In this case $\dim C_j=6,20,24,12,2$ para $j=0,\dots,4$
#
# Here we interpret the matrices of $\delta_j$ as having values in
# $\mathbb{C}[s]$.
#
# The results below state that the ranks of these matrices are:
# $6,14,12,2$, independently of the value of $s\neq 1$.
#
# Therefore, the dimensions of the kernels are: $14,10,2,0$.

# In[ ]:


pr=(s,)
Maux=[A[_,pr] for _ in [4,3,2,1]]
Saux=[mat.smith_form() for mat in Maux]
for mat in Saux:
    aux=[mat[0][j,j] for j in range(min(mat[0].dimensions()))]
    print aux,len([_ for _ in aux if _!=0]),mat[0].dimensions()

# Similarly for $(\zeta,\xi,\mu)=(1,\xi,1)$, con $\xi\neq 1$.

# In[ ]:


pr=(t,)
Maux=[A[_,pr] for _ in [4,3,2,1]]
Saux=[mat.smith_form() for mat in Maux]
for mat in Saux:
    aux=[mat[0][j,j] for j in range(min(mat[0].dimensions()))]
    print aux,len([_ for _ in aux if _!=0]),mat[0].dimensions()

# The following cell examines the $(\zeta,\xi,\mu)=(1,1,\mu)$, with
# $\mu\neq 1$.
#
# In this case $\dim C_j=5,18,23,12,3$ para $j=0,\dots,4$
#
# Here we interpret the matrices $\delta_j$ as having values in
# $\mathbb{C}[u]$.
#
# The results below state that the ranks of these matrices are:
# $5,13,9,3$, independently of the value of $s\neq 1$.
#
# Therefore, the dimensions of the kernels are: $13,10,3,0$.
#
# For this sequence all the homology groups but $H_2$ are trivial. Since
# this happens for $m-1$ roots of unity, we have found an $m-1$-dimensional
```

```
# subspace of $H_2$.

# In[ ]:


pr=(u,)
Maux=[A[_,pr] for _ in [4,3,2,1]]
Saux=[mat.smith_form() for mat in Maux]
for mat in Saux:
    aux=[mat[0][j,j] for j in range(min(mat[0].dimensions()))]
    print aux,len([_ for _ in aux if _!=0]),mat[0].dimensions()

# In[ ]:


def smith2(A):
    dg=[]
    pvt=True
    B=copy(A)
    while pvt and min(B.dimensions())>=0:
        U=B.list()
        U0=[v.degree()==0 for v in U]
        pvt=not prod([not v for v in U0])
        if not pvt:
            return [dg,B]
        k=ZZ(U0.index(True))
        i,j=k.quo_rem(B.ncols())
        B.swap_rows(0,i)
        B.swap_columns(0,j)
        for i in range(1,B.nrows()):
            B.add_multiple_of_row(i,0,-B[i,0]/B[0,0])
        for j in range(1,B.ncols()):
            B.add_multiple_of_column(j,0,-B[0,j]/B[0,0])
        dg.append(B[0,0])
        B=B.delete_rows([0]).delete_columns([0])

# In[ ]:


def smith2var(A,pr):
    dg,B=smith2(A)
    an=anillos[pr]
    for i in range(B.nrows()):
        Bi=B[i]
        mcd=gcd(Bi.list())
        for vv in pr:
            if an(vv-1).divides(mcd):
                Bi1=Bi.change_ring(an.fraction_field())
                Bi2=Bi1/an(vv-1)
                Bi=Bi2.change_ring(an)
        B[i]=Bi
    for i in range(B.ncols()):
```

```
        Bi=B.column(i)
        mcd=gcd(Bi.list())
        for vv in pr:
            if an(vv-1).divides(mcd):
                Bi1=Bi.change_ring(an.fraction_field())
                Bi2=Bi1/an(vv-1)
                Bi=Bi2.change_ring(an)
        for j in range(B.nrows()):
            B[j,i]=Bi[j]
    dg1,B1=smith2(B)
    return dg+dg1,B1

# The following cell examines the $(\zeta,\xi,\mu)=(\zeta,\xi,1)$, with
# $\zeta,\xi\neq 1$.
#
# In this case $\dim C_j=4,17,23,12,2$ para $j=0,\dots,4$
#
# Here we interpret the matrices $\delta_j$ as having values in
# $\mathbb{C}[s,t]$ for which the Smith form does not need to exist,
# but we can apply elementary operations.
#
# The results below state that the ranks of these matrices are:
# $4,13,10,2$, independently of the value of $s,t\neq 1$.
#
# Something similar happens for the cases $(\zeta,\xi,\mu)=(\zeta,1,\mu)$, with
# $\zeta,\mu\neq 1$ and $(\zeta,\xi,\mu)=(1,\xi,\mu)$, with $\xi,\mu\neq 1$.

# In[ ]:


pr=(s,t)
Maux=[A[_,pr] for _ in [1..4]]
for i in [1..4]:
    n0,m0=Maux[i-1].dimensions()
    print "******************************"
    dg,SM=smith2(Maux[i-1])
    dg1,SM1=smith2var(SM,pr)
    print "Rango de M",i-1,": ",m0
    print "Rango de M",i,": ",n0
    print "diagonalizado en delta",i,": ",dg+dg1,len(dg+dg1)
    if SM1!=0:
        print "parte no diagonalizada de delta",i,": ",show(SM1)
    print "******************************"
    print "\n"

# In[ ]:


pr=(s,u)
Maux=[A[_,pr] for _ in [1..4]]
```

```
for i in [1..4]:
    n0,m0=Maux[i-i].dimensions()
    print "********************************"
    dg,SM=smith2(Maux[i-1])
    dg1,SM1=smith2var(SM,pr)
    print "Rango de M",i-1,": ",m0
    print "Rango de M",i,": ",n0
    print "diagonalizado en delta",i,": ",dg+dg1,len(dg+dg1)
    if SM1!=0:
        print "parte no diagonalizada de delta",i,": ",show(SM1)
    print "********************************"
    print "\n"

# In[ ]:


pr=(t,u)
Maux=[A[_,pr] for _ in [1..4]]
for i in [1..4]:
    n0,m0=Maux[i-i].dimensions()
    print "********************************"
    dg,SM=smith2(Maux[i-1])
    dg1,SM1=smith2var(SM,pr)
    print "Rango de M",i-1,": ",m0
    print "Rango de M",i,": ",n0
    print "diagonalizado en delta",i,": ",dg+dg1,len(dg+dg1)
    if SM1!=0:
        print "parte no diagonalizada de delta",i,": ",show(SM1)
    print "********************************"
    print "\n"-1, -1, -1, -1

# The following cell examines the $(\zeta,\xi,\mu)=(\zeta,\xi,\mu)$, with
# $\zeta,\xi,\mu\neq 1$.
#
# In this case $\dim C_j=1,12,21,12,2$ para $j=0,\dots,4$
#
# Here we interpret the matrices $\delta_j$ as having values in
# $\mathbb{C}[s,t,u]$ for which the Smith form does not need to exist,
# # but we can apply elementary operations.
#
# The results below state that the ranks of these matrices are:
# $\delta_1:1$, 4, $\delta_3:3$, $\delta_4:2$, independently of
# the value of $s,t,u\neq 1$. But $\delta_2$ has rank 10 if $t=s$ and rank
# 11 if $t\neq s$. This happens in $(d-1)(m-1)$ cases, $d=\gcd(a,b)$.
#
# It can be calculated that $H_2,H_1$ has rank $1$ if $s=t$; and zero
# in any other case.

# In[ ]:


pr=(s,t,u)
```

```
Maux=[A[_,pr] for _ in [1..4]]
for i in [1..4]:
    n0,m0=Maux[i-1].dimensions()
    print "******************************"
    dg,SM=smith2(Maux[i-1])
    dg1,SM1=smith2var(SM,pr)
    print "Rango de M",i-1,": ",m0
    print "Rango de M",i,": ",n0
    print "diagonalizado en delta",i,": ",dg+dg1,len(dg+dg1)
    if SM1!=0:
        print "parte no diagonalizada de delta",i,": "
        show(SM1)
    print "******************************"
    print "\n"
```