Capítulo 1

Codificación topológico/combinatoria de diagramas planos

Este primer Capítulo de la memoria está dedicado al estudio de cómo codificar toda la información topológico/combinatoria de lo que denominamos *diagramas*. Un diagrama será esencialmente la inmersión de un grafo en una cierta superficie compacta (o en el plano afín). En consecuencia, la noción de diagrama engloba a las dos que ocuparán el resto de la memoria, que son las curvas algebraicas planas reales y los diagramas de Voronoi o de Delaunay.

La herramienta fundamental para la codificación serán las *listas de aristas* que se definirán en la Sección 1.2. Probaremos que dichas listas codifican toda la información de un entorno del diagrama. La demostración se basa en la existencia de ciertos entornos del diagrama (los *entornos regulares*) la cual a su vez está basada en el hecho de que los diagramas sean triangulables. En otras palabras: aunque no exigimos a los diagramas ninguna propiedad geométrica específica, puede demostrarse que su estructrura coincide siempre con la de algún diagrama que esté formado por unión finita de segmentos rectilíneos en una superficie poliedral. Ésto nos permitirá trabajar en la categoría PL y, de hecho, es lo que hace que la topología de los objetos se pueda caracterizar por métodos combinatorios.

Las listas de aristas que utilizamos guardan una cierta relación con las estructuras de datos que habitualmente se usan para codificar la inmersión de un grafo en el plano. Estas estructuras constan de una lista de las aristas del grafo dando, para cada arista e, una cierta información que nos dice qué otras aristas la siguen en sentido horario y/o antihorario en cada uno de sus extremos. Hemos de citar en este sentido a [Baumgart], [Guibas-Stolfi], [Muller-Prep.] y [Preparata-Sha.]. En algunos casos se añade alguna información adicional (por ejemplo, sobre las caras del diagrama) que, aunque es redundante, hace la estructura de datos más operativa. Si la superficie no es orientable (lo cual se admite en [Guibas-Stolfi]), se requiere un cuidado especial ya que los sentidos "horario" y "antihorario" no tienen significado global. La diferencia esencial entre ese punto de vista y el nuestro es que lo que a nosotros nos interesa no es tanto la efectividad de la estructura de datos para realizar sobre ella algoritmos geométricos de manera rápida, sino la posibilidad de

decidir si las inmersiones representadas por dos de estas estructuras son equivalentes o no.

Por otra parte, nuestro principal objetivo al emprender este estudio era su aplicación al caso de curva algebraicas, lo cual nos obliga a una cierta generalidad. Por ejemplo, en todas las referencias mencionadas arriba, el diagrama $\mathcal{D}$ se supone conexo y sus caras simplemente conexas. Para curvas algebraicas, tenemos que admitir la posibilidad de trabajar en el caso no conexo. Una vez que admitimos ésto, nuestros resultados resultan ser válidos para un caso bastante más general que el que contaría sólo con los diagramas producidos por curvas algebraicas.

Debemos mencionar que nuestras listas tienen un referente lejano también en las que aparecen en la tesis de Antonio González-Corbalán (cf. [G.Corbalán] o [G.Corbalán-Recio]), aunque aquí la similitud es más en la filosofía del planteamiento que en el aspecto de la solución.

La organización del Capítulo es como sigue:

La Sección 1.1 está dedicada a definir con precisión lo que entendemos por diagramas y por equivalencia de los mismos y a introducir la herramienta técnica en la que se basa nuestro método, que son los ya mencionados entornos regulares.

En la siguiente introducimos las listas de aristas, como forma de codificar el entorno regular de un diagrama. En la definición de la lista de aristas de un diagrama se hacen varias elecciones arbitrarias, por lo cual un mismo diagrama puede producir diversas listas diferentes (aunque la longitud de la lista sí queda fijada por el diagrama). Para solventar ésto definiremos una *lista canónica* de entre todas las posibles, en la Sección 1.2.2. La forma de definir esta lista canónica (como la más pequeña posible respecto a un cierto orden lexicográfico) no encierra ningún misterio. Lo que sí resulta notable es que la obtención de la lista canónica pueda realizarse de una manera muy rápida (en tiempo cuadrático respecto a la longitud de la lista) habida cuenta de que el número de listas posibles para un diagrama es exponencial. Terminamos la Sección mostrando que la lista de aristas es suficiente para caracterizar, módulo equivalencia, aquéllos diagramas cuyas caras sean todas celdas.

En la Sección 1.3 abordamos la forma de codificar completamente el diagrama, añadiendo información a la lista. Más bien, lo que se hace es dividir la lista en "componentes conexas" que se corresponden con las del diagrama y disponer éstas en una estructura de grafo que representa cómo las componentes del diagrama van unidas a sus caras. Con un poco de información adicional ésto caracterizará el diagrama. De nuevo nos interesaremos en encontrar una forma canónica de organizar toda esta información (a la que llamamos el *código* el diagrama). El problema resulta ahora más complicado porque conlleva en particular decidir si los *grafos de componentes* de dos diagramas son isomorfos y cualquier grafo bipartito puede aparecer como tal. Puesto que el problema de isomorfismo de grafos no tiene solución conocida en tiempo polinomial, el nuestro tampoco puede esperar tenerla.

Sin embargo, en los casos más interesantes (cuando la superficie sea una esfera, el plano afín o el plano proyectivo) el grafo de componentes será un árbol. En este caso sí que podemos dar un código canónico en el mismo tiempo asintótico que necesitábamos para la lista canónica (cf. 1.3.2).

El capítulo termina con una sección dedicada a describir una implementación de nuestro algoritmo, aplicado a curvas algebraicas planas reales afines. La imple-

mentación se ha realizado como extensión de un programa que calcula la topología de una curva algebraica por medio de una descomposición cilíndrica del plano, producido por Hoon Hong, del RISC-Linz. Está escrita en C utilizando la librería SACLIB y en ella ha colaborado Mark J. Encarnación, del mismo instituto. En un apéndice se dan las funciones utilizadas por el algoritmo de codificación.

1.1 Diagramas. Primeras propiedades.

1.1.1 Definiciones

A lo largo de este Capítulo S será una superficie compacta, i.e. un espacio topológico compacto y Hausdorff localmente homeomorfo al plano $\mathbb{R}^2$ en todos sus puntos. En ocasiones trabajaremos con el propio plano $\mathbb{R}^2$ o con superficies con borde, i.e. espacios topológicos compactos y Hausdorff que en todos sus puntos son localmente homeomorfos o bien al plano o bien a un semiplano cerrado.

Vamos a introducir la clase de objetos fundamentales en nuestra memoria, a los que llamaremos *diagramas*. Un diagrama es esencialmente la inmersión de un grafo en una superficie compacta. Sin embargo, preferiremos utilizar el término "diagrama" porque nuestro interés primordial no está en su estructura como grafos, sino en su topología como subconjuntos de la superficie. Es por ello que admitiremos la existencia de bucles y multi-aristas en el grafo.

Definición 1.1.1 *Un* pseudo-grafo G *(cf. [Harary]) es un grafo al que se le permite tener* multi-aristas *(aristas diferentes que conectan el mismo par de vértices) y* bucles *(aristas que no conectan dos vértices diferentes, sino que tienen al mismo en sus dos extremos.*

Definición 1.1.2 *Llamaremos* diagrama $\mathcal{D} : G \mapsto S$ *a la inmersión de un pseudo-grafo en una superficie compacta S.*

Llamaremos caras *del diagrama a las componentes conexas de $S \setminus \mathcal{D}(G)$. Llamaremos* aristas *y* vértices *del diagrama a los del pseudo-grafo G.*

La imagen *del diagrama será la imagen $\mathcal{D}(G)$ del grafo G en la superficie S.*

El hecho de que exijamos compacidad a las superficies en las que se trabajamos se debe esencialmente a la existencia de un buen teorema de clasificación de superficies compactas y a la comodidad de trabajar en espacios compactos. En realidad, nuestras técnicas serían adaptables a superficies no compactas pero que admitan una fácil caracterización topológica (por ejemplo, a todas las superficies que resultan de eliminar un número finito de puntos de una superficie compacta, con o sin borde). Sin embargo, la única superficie no compacta a la que haremos referencia en esta memoria será el plano Euclídeo. Para casi todos los efectos el plano Euclídeo será considerado una esfera con un punto distinguido (el "infinito").

Para el plano Euclídeo la definición de diagrama es la siguiente.

Definición 1.1.3 *Un pseudo-grafo* abierto G^* *será un pseudo-grafo G al que se le ha eliminado uno de sus vértices y tiene, por tanto, un cierto número finito de aristas "abiertas". Llamaremos* diagrama *en el plano Euclídeo a toda inmersión $\mathcal{D}^* : G^* \mapsto \mathbb{R}^2$ cerrada de un cierto pseudo-grafo abierto G^* en el plano Euclídeo $\mathbb{R}^2$. Esto es equivalente a decir que $\mathcal{D}^*$ puede prolongarse a una inmersión $\mathcal{D}$:*

$G \mapsto S^2$, donde $S^2 = \mathbb{R}^2 \cup \{\infty\}$ es una esfera, ∞ es el punto del infinito y G es el pseudo-grafo que se obtiene como compactificación de G^* por un punto.

Es decir, un diagrama en el plano será esencialmente la misma cosa que una diagrama en la esfera con un cierto punto distinguido, el cual juega el papel de punto del infinito. La Figura 1.1 muestra el diagrama de un "cubo" en una esfera (parte (a)) y dos diagramas planos que se obtienen de él tomando como punto del infinito uno en una cara (parte (b)) y otro en un vértice (parte (c)).

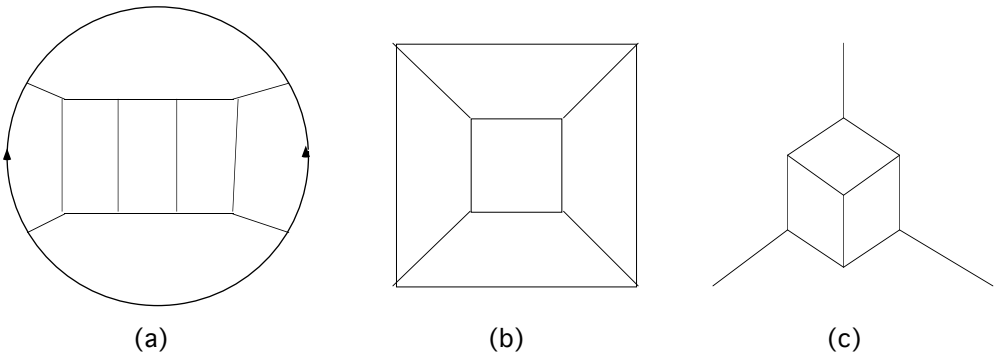


Figura 1.1: Ejemplos de diagramas.

Para simplificar la notación normalmente identificaremos G con su imagen $\mathcal{D}(G)$ en S, y entenderemos las aristas y vertices del diagrama como partes de $\mathcal{D}(G)$, es decir, como inmersas en la superficie S. De esta forma todo diagrama induce una descomposición de la superficie S en vértices, aristas y caras, con un número finito de cada uno de ellos. Las aristas son homeomorfas a rectas, ya que son 1-variedades no-compactas (poseen al menos un vértice en su adherencia). Como el conjunto $\mathcal{D}(G)$ es cerrado en S las caras son abiertas y son, por tanto, variedades 2-dimensionales. En la Proposición 1.1.9 veremos que cada una es homeomorfa al interior de una superficie con borde.

La imagen $\mathcal{D}(G)$ de un diagrama, considerada como espacio topológico, posee en sí misma una estructura *natural* de pseudo-grafo, independiente del pseudo-grafo G que lo define. La siguiente definición nos indica cómo construir este pseudo-grafo *natural* asociado a $\mathcal{D}(G)$. Por esta razón, cuando sólo nos interese la topología del diagrama podremos, sin ambigüedad, abusar del lenguaje y llamar diagrama a la imagen $\mathcal{D}(G)$ de G en S por la inmersión $\mathcal{D}$, en vez de a la inmersión en sí.

Definición 1.1.4 *Sea $\mathcal{D} : G \mapsto S$ un diagrama en una superficie compacta. El* pseudo-grafo *natural asociado al diagrama se define de la siguiente manera.*

Supongamos en primer lugar que $\mathcal{D}(G)$ es conexo. Si $\mathcal{D}(G)$ es localmente homeomorfo a una recta en todo punto, entonces $\mathcal{D}(G)$ es una 1-variedad topológica compacta y es, por tanto, homeomorfo a un círculo. Su estructura natural de pseudo-grafo es la de un bucle cuyo único vértice es un punto arbitrario de $\mathcal{D}(G)$. Si, en cambio, hay puntos donde $\mathcal{D}(G)$ no es localmente homeomorfo a una recta (a los cuales llamaremos puntos topológicamente singulares*), entonces la estructura natural de pseudo-grafo de $\mathcal{D}(G)$ se obtiene tomando como vértices precisamente esos puntos singulares, y como aristas las componentes conexas de $\mathcal{D}(G) \setminus V$, siendo V el conjunto de vértices.*

Para un diagrama no conexo, la estructura natural de pseudo-grafo es la unión disjunta de los pseudo-grafos naturales asociados a sus componentes conexas.

Dicho de otra forma, la obtención del pseudo-grafo natural asociado a la imagen del diagrama a partir del pseudo-grafo original que lo define se hace uniendo entre sí las dos aristas incidentes a cada vértice de valencia dos, excepto cuando dicho vértice sea el único de una componente conexa. Por tanto, dado un cierto diagrama $\mathcal{D} : G \mapsto S$, entonces G es el pseudo-grafo natural asociado a $\mathcal{D}$ si y solo si los únicos vértices de G que tienen valencia dos están unidos a un bucle. La Figura 1.2 muestra un cierto diagrama en el plano proyectivo y su estructura natural de pseudo-grafo.

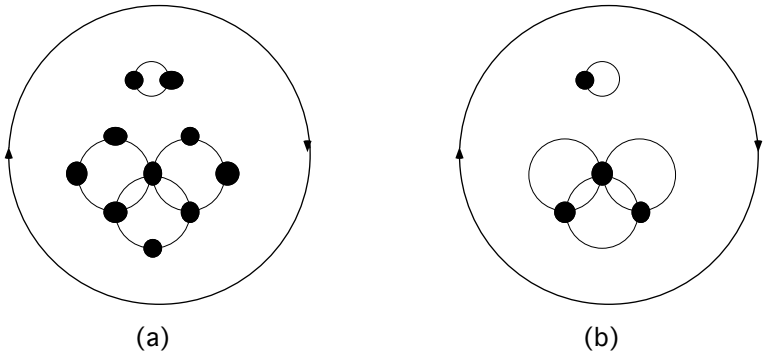


Figura 1.2: Un diagrama y su pseudo-grafo natural.

1.1.2 Entornos regulares

Nuestras técnicas de este Capítulo estarán basadas en la existencia de *entornos regulares* de los diagramas con los que trabajamos. Ésta a su vez se basa en la *triangulabilidad* de los mismos.

Definición 1.1.5 *Una* triangulación *de un espacio topológico E es un homeomorfismo $h : |K| \to E$ entre el soporte de un cierto complejo simplicial K y el espacio E. Dada una inmersión $\mathcal{D} : L \to E$ de un cierto complejo simplicial en E, una triangulación h de E se dice* compatible *con $\mathcal{D}$ si la imagen de cada símplice de L por $\mathcal{D}$ es a su vez imagen de un cierto símplice de K por h.*

La definición de triangulación compatible se aplica a los diagramas definidos como inmersiones de pseudo-grafos puesto que todo pseudo-grafo se puede convertir en un complejo simplicial (i.e. en un grafo) por subdivisión de algunas de sus aristas.

Definición 1.1.6 *Sea $\mathcal{D} : G \to S$ un diagrama en una superficie S. Diremos que una triangulación $h : |K| \to S$ de S es* compatible *con el diagrama $\mathcal{D}$ si existe una subdivisión de G que lo convierte en un grafo G' y tal que cada imagen de una arista o vértice de G' por $\mathcal{D}$ es una arista o vértice de la triangulación.*

Todo diagrama es *triangulable* en el sentido de que existe una triangulación de S compatible con él. Sin embargo, una propiedad análoga en espacios tridimensionales no es cierta. Por ejemplo, existen inmersiones de segmentos en $\mathbb{R}^3$ que no son

compatibles con ninguna triangulación de un entorno de los mismos (cf. [Moise] o [Fox-Artin]). Para el caso de dimensión dos, Moise [Moise, Teorema 10.13] demuestra que dado la inmersión de un complejo simplicial L en $\mathbb{R}^2$, siempre existe una triangulación de $\mathbb{R}^2$ compatible con una subdivisión de L. Aunque ésto no se aplica directamente a nuestro caso, por trabajar en superficies diferentes, su demostración puede adaptarse literalmente (de hecho, el trabajar en espacios compactos simplifica las cosas porque nuestros complejos simpliciales serán siempre finitos, mientras que a los de [Rourke-Sand.] sólo se les exige ser localmente finitos). Por otra parte, la tesina de licenciatura del autor [Santos1, Teorema 2.5.1], contiene también una demostración de que todo diagrama es triangulable.

Sea $\mathcal{D} : G \to S$ un cierto diagrama en una superficie S. Dada una triangulación $h : |K| \to S$ de S compatible con $\mathcal{D}$, llamaremos símplices, vértices, aristas y triángulos de la triangulación a las imágenes de los símplices, vértices, aristas y triángulos de K por h. La unión de todos los triángulos que tienen intersección no vacía con la imagen $\mathcal{D}(G)$ del diagrama es un cierto entorno N de $\mathcal{D}(G)$. Si satisface unas ciertas condiciones lo llamaremos regular:

Definición 1.1.7 *Sea $\mathcal{D} : G \to S$ un cierto diagrama en una superficie S. Sea $h : |K| \to S$ una triangulación de S compatible con $\mathcal{D}$. Sea N el entorno de la imagen de $\mathcal{D}$ formado por todos los triángulos de la triangulación que tienen intersección no vacía con dicha imagen. Diremos que N es un* entorno regular *de $\mathcal{D}(G)$ (o, por abuso del lenguaje, del diagrama $\mathcal{D}$) si*

(i) N es una superficie compacta con borde.

(ii) La intersección de cualquier símplice de la triangulación con $\mathcal{D}$ es vacía o una cara de dicho símplice.

(iii) El borde de N está formado exactamente por aquéllas aristas y vértices de la triangulación que tienen intersección vacía con $\mathcal{D}(G)$ pero están contenidos en un triángulo que tiene intersección no vacía con $\mathcal{D}(G)$.

Un entorno N arbitrario de la imagen de $\mathcal{D}$ se dirá regular *si es el entorno regular definido por alguna triangulación de S compatible con $\mathcal{D}$. Llamaremos* complemento regular *de $\mathcal{D}$ a la clausura del complementario de un entorno regular y* caras regulares *a las componentes conexas de un complemento regular.*

La definición 1.1.7 proviene del Teorema 3.11 de [Rourke-Sand.]. Aunque la definición que allí se da de entorno regular se aplica a un contexto más general, el mencionado teorema (que recibe el nombre de *Teorema del Entorno Simplicial*) es una caracterización de los entornos regulares de subespacios PL de una variedad. La Figura 1.3 muestra dos triangulaciones compatibles con un mismo diagrama. La primera (parte (a) de la Figura) define un entorno regular. La segunda no: el entorno no es una superficie con borde, hay dos triángulos cuya intersección con el diagrama es disconexa y hay una arista que no corta al diagrama, contenida en dos triángulos que sí lo cortan y que no está en el borde del entorno.

Todo diagrama posee entornos regulares. Para conseguirlos basta con refinar suficientemente una triangulación arbitraria compatible con el diagrama. Por ejemplo, dado cualquier subcomplejo simplicial L de un complejo simplicial K, la segunda

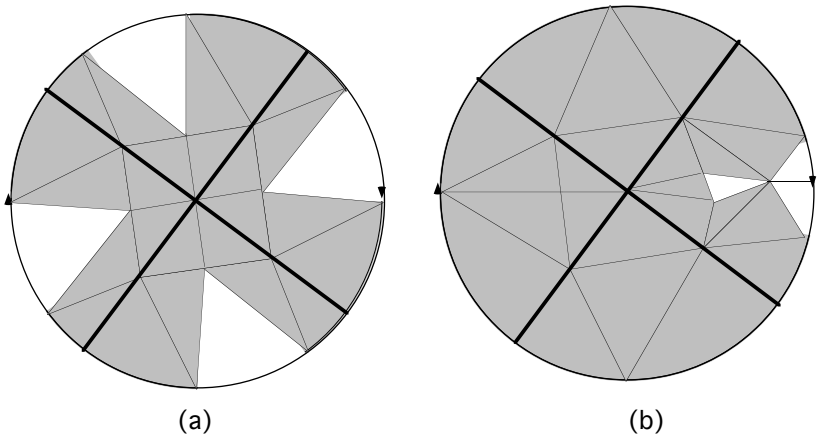


Figura 1.3: Un entorno regular de un diagrama y uno que no lo es.

subdivisión baricéntrica bbK de$K$ (que es la subdivisión baricéntrica de la subdivisión baricéntrica bK de K) define un entorno regular de L (cf. [Moise]). La figura 1.4 nos muestra un ejemplo de lo que ésto significa. Una consecuencia de ésto es que los entornos regulares de $\mathcal{D}(G)$ en S forman una base de entornos de $\mathcal{D}$, ya que la triangulación K a partir de la cual se obtiene la segunda subdivisión baricéntrica se puede tomar tan fina como se quiera.

Otra propiedad esencial de los entornos regulares es su unicidad topológica: si N_L y $N_{L'}$ son dos entornos regulares de $\mathcal{D}(G)$ en S para dos triangulaciones diferentes L y L' de S compatibles con $\mathcal{D}$, entonces existe una isotopía ambiente de S en sí mismo que deja fijo a $\mathcal{D}(G)$ y envía N_L sobre $N_{L'}$ [Rourke-Sand., 3.24]. Ésto implica el mismo tipo de unicidad para el complementario, es decir, lo que nosotros hemos llamado *complemento regular* para el caso de un diagrama. Otras propiedades de los entornos regulares son:

Proposición 1.1.8 *Sea $\mathcal{D} : G \mapsto S$ un diagrama. Sea N un entorno regular de $\mathcal{D}$, M su complemento regular.*

(i) N y M son superficies con borde [Rourke-Sand., Prop. 3.10, Cor. 3.14]. El borde de ambas es común y coincide con su intersección. Lo designaremos por $\dot{N}$.

(ii) Si $N(T)$ y $N'(T)$ son dos entornos regulares de T en S, el primero contenido en el interior del segundo, entonces la clausura de $N'(T)\backslash N(T)$ es homeomorfa a $\dot{N}(T) \times I$, donde I es un intervalo [Rourke-Sand., 3.18].

(iii) La inclusión $\mathcal{D}(G) \to N$ es una equivalencia homotópica [Rourke-Sand., Cor. 3.30]. □

Finalmente, demostremos algunas propiedades más que vamos a necesitar:

Proposición 1.1.9 *Sea $\mathcal{D} : G \mapsto S$ un diagrama. Sea N un entorno regular de $\mathcal{D}$, M su complemento regular.*

(i) Cada componente conexa de N contiene una y sólo una componente conexa de $\mathcal{D}(G)$.

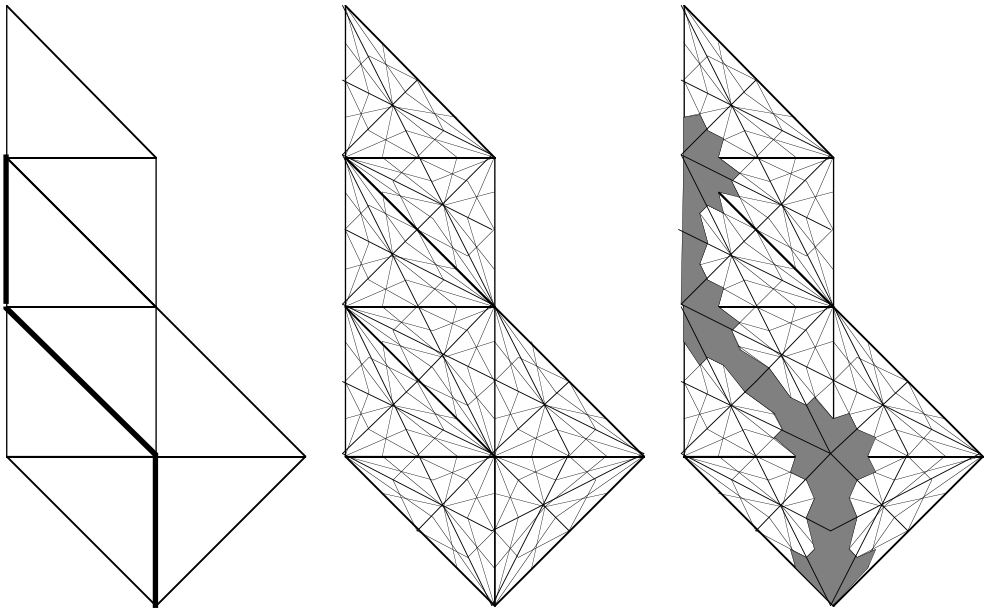


Figura 1.4: Entorno regular obtenido de la segunda subdivisión baricéntrica.

(ii) Cada componente conexa de $N \setminus \mathcal{D}(G)$ *es homeomorfa al producto de una circunferencia y un intervalo semiabierto* $[0,1)$.

(iii) Cada componente conexa de $S \setminus \mathcal{D}(G)$ *(es decir, cada cara del diagrama) contiene una y sólo una componente conexa de* M *(es decir, una cara regular).*

(iv) El interior de una cara regular es homeomorfo a la cara del diagrama en la que está contenida.

Demostración: Sea K la triangulación que define el entorno regular N. Sin ambigüedad podemos identificar cada símplice de K con su imagen en la superficie S y hablar de los triángulos, aristas y vértices de K como inmersos en ella.

(i) Toda componente conexa de N contiene al menos una de $\mathcal{D}(G)$, puesto que todo triángulo de K que forma parte de N corta a $\mathcal{D}(G)$. Supongamos, por reducción al absurdo, que una cierta componente N_0 de N contiene dos componentes conexas distintas de $\mathcal{D}(G)$. Consideremos una cadena de triángulos $T_1, T_2, \ldots, T_l$ de triángulos de K contenidos en N_0 tal que dos de ellos consecutivos sean adyacentes por una arista y tal que T_1 y T_l corten a componentes conexas diferentes de $\mathcal{D}(G)$. Supongamos además que l es el mínimo número de triángulos en esas condiciones y lleguemos a una contradicción. Si $l \geq 3$, consideremos el triángulo T_2. Como todo triángulo de N_0 tiene intersección no vacía con $\mathcal{D}(G)$, en particular T_2 la tiene, así que o bien T_1, T_2 o $T_2, \ldots, T_l$ es una cadena en las mismas condiciones, lo cual nos proporciona una de longitud menor. Si $l = 1$ la contradicción es que entonces $T_1 \cap \mathcal{D}(G)$ no puede ser una cara de T_1 y por tanto K no satisface la condición (ii) de la definición de entorno regular. Por último, si $l = 2$ la única posibilidad es que la arista común de T_1 y T_2 no corte a $\mathcal{D}(G)$. La condición (iii) de la definición de entorno regular implicaría que dicha arista está en el borde de N_0, lo cual es imposible.

(ii) De la definición de entorno regular se deduce que cada triángulo de N tiene o bien un vértice en el borde de N y la arista opuesta en $\mathcal{D}(G)$ o bien un vértice en $\mathcal{D}(G)$ y la arista opuesta en el borde de N. Ésto, unido al hecho de que N es una variedad con borde, implica la afirmación.

(iii) y (iv) Sea F una cara del diagrama. La unión de las caras con borde que contiene forman una superficie compacta con borde y, por la propiedad (ii), la cara F se obtiene pegando a cada componente del borde de dicha superficie un cilindro homeomorfo a $S^1 \times [0,1)$. El resultado de hacer esta operación a una superficie con borde compacta arbitraria B es siempre una superficie homeomorfa al interior de B, lo cual prueba las dos afirmaciones. □

1.1.3 Equivalencia de diagramas

El objetivo último de este capítulo es el estudio de cuando las descomposiciones (en caras, aristas y vértices) inducidas por dos diagramas son combinatoriamente iguales. Una definición precisa de este concepto puede hacerse en términos topológicos:

Definición 1.1.10 *Sean $\mathcal{D}_1 : G_1 \mapsto S_1$ y $\mathcal{D}_2 : G_2 \mapsto S_2$ sendos diagramas. Diremos que $\mathcal{D}_1$ y $\mathcal{D}_2$ son* combinatoriamente equivalentes *(o, simplemente,* equivalentes*) si existe un homeomorfismo $h : S_1 \longrightarrow S_2$ de S_1 en S_2 que envía biyectivamente la imagen de $\mathcal{D}_1$ sobre la imagen de $\mathcal{D}_2$ y los vértices de $\mathcal{D}_1$ sobre vértices de $\mathcal{D}_2$. Si las superficies S_1 y S_2 son* orientables *y nos han sido dadas con una orientación fijada, normalmente exigiremos que el homeomorfismo h* conserve la orientación*.*

La exigencia de que el homeomorfismo conserve la orientación es, en cierto modo, opcional. Nuestros métodos se adaptan bien al caso en que nos interesa testar "equivalencia conservando la orientación" porque están basados en elegir una orientación local en un entorno de cada vértice de un diagrama. Para diagramas en superficies orientables, el poder elegir las orientaciones de forma compatible (es decir, elegir una orientación global de la superficie) simplifica un tanto las cosas. En cualquier caso, si quisiéramos testar equivalencia de dos diagramas $\mathcal{D}_1$ y $\mathcal{D}_2$ sin exigir a la equivalencia que conserve la orientación, bastaría con testar la equivalencia (conservando la orientación) de $\mathcal{D}_1$ con $\mathcal{D}_2$ primero y con una imagen especular de $\mathcal{D}_2$ después.

La definición 1.1.10 es válida también para el caso de diagramas en el plano. En este caso, dos diagramas en el plano son equivalentes de acuerdo con ella si y sólo si los correspondientes diagramas en la esfera lo son y el homeomorfismo que envía una esfera sobre la otra se puede elegir de forma que, además de enviar un diagrama sobre el otro y los vértices de uno sobre los del otro, envíe el punto del infinito al punto del infinito.

Al objeto de estudiar la equivalencia de dos diagramas lo que haremos será diseñar una manera de "codificar" todas las propiedades de cada uno de ellos que son relevantes para la equivalencia y después testar la equivalencia en los códigos correspondientes. El "código" asociado a un diagrama será por tanto una serie (finita) de símbolos obtenible a partir del diagrama y que verifique que dos diagramas con el mismo código sean equivalentes. Ésto se hará en las secciones 1.2 y 1.3. Varias maneras de hacerlo existen en la bibliografía ([G.Corbalán], [G.Corbalán-Recio],

[Guibas-Stolfi], [Muller-Prep.], pero ninguna tan general como la nuestra. Nótese, por ejemplo, que nuestros diagramas no son necesariamente conexos (lo cual se exige implícita o explícitamente en todas las referencias mencionadas) ni sus caras simplemente conexas.

La codificación de diagramas no será, en principio, unívoca. En la codificación se harán varias elecciones arbitrarias que hagan que un mismo diagrama pueda dar lugar a varios códigos diferentes. Lo que sí tendremos al menos será una noción de "equivalencia de códigos" y una relación biunívoca entre las clases de equivalencia de diagramas y las clases de equivalencia de códigos. Además, cada clase de equivalencia de códigos tendrá una cantidad finita de elementos, con lo cual siempre será posible testar si dos códigos son equivalentes comparando uno de ellos con todos los de la clase de equivalencia del otro. Sin embargo, lo que nosotros proponemos es algo más eficiente. A saber, elegir, de entre cada clase de equivalencia, un código en particular (al que llamaremos *canónico*) que se pueda obtener de forma relativamente sencilla a partir de cualquier código de la clase. En los casos más interesantes (en que el diagrama esté inmerso en la esfera, el plano real o el plano proyectivo) el código canónico se obtendrá en tiempo casi cuadrático en la *complejidad intrínseca* del diagrama, lo cual supone una importante mejora respecto a la enumeración de todos los códigos, que son exponencial en número. Para superficies arbitrarias el algoritmo será un poco peor pero aún polinomial en la complejidad del diagrama, una vez fijada la superficie.

Mención especial merece la equivalencia topológica que, en realidad, constituye nuestra motivación principal.

Definición 1.1.11 *Sean $\mathcal{D}_1 : G_1 \mapsto S_1$ y $\mathcal{D}_2 : G_2 \mapsto S_2$ sendos diagramas. Diremos que $\mathcal{D}_1$ y $\mathcal{D}_2$ son* topológicamente equivalentes *(o que tienen el mismo* tipo topológico*) si existe un homeomorfismo $h : S_1 \longrightarrow S_2$ de S_1 en S_2 que envía biyectivamente la imagen de $\mathcal{D}_1$ sobre la imagen de $\mathcal{D}_2$. Si las superficies S_1 y S_2 son* orientables*, y nos han sido dadas con una orientación fijada, normalmente exigiremos que el homeomorfismo h* conserve la orientación*.*

Es claro que dos diagramas que sean equivalentes de acuerdo con la definición 1.1.10 serán también topológicamente equivalentes. El recíproco no es cierto. La diferencia entre las dos definiciones es que en la definición inicial de equivalencia se exige que el homeomorfismo envíe vértices de un diagrama a vértices del otro. En la nueva definición, ésta exigencia se mantiene implícitamente para vértices de valencia diferente de dos (es decir, para puntos topológicamente singulares) pero desaparece para los vértices de valencia dos, que no afectan a la topología del diagrama.

Dicho de otra manera, en la equivalencia topológica estamos considerando los diagramas como subconjuntos de la superficie en que están inmersos, olvidándonos del pseudo-grafo del que provienen. Por esta razón, la equivalencia topológica es lo mismo que equivalencia de los diagramas definidos por los pseudo-grafos naturales.

Proposición 1.1.12 *Sean $\mathcal{D}_1 : G_1 \mapsto S_1$ y $\mathcal{D}_2 : G_2 \mapsto S_2$ sendos diagramas. Entonces son equivalentes:*

(i) $\mathcal{D}_1$ y $\mathcal{D}_2$ son topológicamente equivalentes.

(ii) Los diagramas $\mathcal{D}'_1 : G'_1 \mapsto S_1$ *y* $\mathcal{D}'_2 : G'_2 \mapsto S_2$ *con la misma imagen que* $\mathcal{D}_1$ *y* $\mathcal{D}_2$ *pero en los cuales* G'_1 *y* G'_2 *son los pseudo-grafos naturales de* $\mathcal{D}_1$ *y* $\mathcal{D}_2$ *son equivalentes.* □

1.2 Codificación de entornos regulares. Listas de aristas

El primer paso para la codificación de un cierto diagrama será construir una lista de aristas del mismo, de la siguiente manera:

1.2.1 Listas de aristas

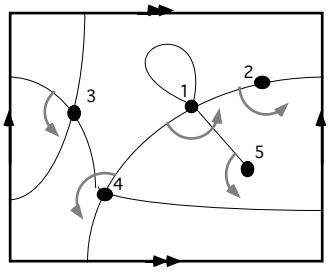
Sea $\mathcal{D} : G \to S$ un cierto diagrama en una superficie S. Sea P uno de sus vértices y sea sea m la valencia de P en el pseudo-grafo. Existe entonces un cierto entorno U del punto P en S y un homeomorfismo de U en un círculo que envía $U \cap \mathcal{D}(G)$ sobre la unión de m radios del círculo. Llamaremos *semiaristas* a las imágenes inversas de dichos radios, que son porciones conexas de cada una de las aristas incidentes en P.

En realidad, una definición más precisa de lo que entendemos por semiarista debería pasar por considerar todos los posibles entornos de i en las condiciones señaladas y llamar semiarista a una clase de equivalencia de porciones conexas de la misma arista (de forma parecida a como se define un gérmen de funciones en teoría de haces). Sin embargo, no creemos necesaria la introducción de tal formalismo. De forma intuitiva, una semiarista será cada una de las dos porciones en los extremos de una arista. Dado que admitimos la posibilidad de que las aristas sean bucles, puede ocurrir que las dos semiaristas de una cierta arista sean incidentes en un mismo vértice.

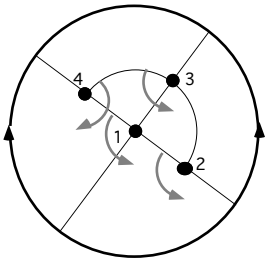
Supongamos ahora que el diagrama $\mathcal{D}$ tiene n vértices, que consideraremos etiquetados de manera arbitraria con los n primeros números naturales $\{1, \ldots, n\}$. Sea m_i la multiplicidad de cada vértice $i = 1, \ldots, n$. Elijamos una cierta orientación local en un entorno de cada vértice i y numeremos las m_i semiaristas incidentes en él siguiendo dicha orientación, aunque comenzando en una de ellas elegida de forma arbitraria. De ésta forma, cada semiarista del diagrama está caracterizada por dos números $i \in \{1, \ldots, n\}$ y $b \in \{1, \ldots, m_i\}$ que representan a su vértice y a la posición de la semiarista alrededor del vértice. De la misma manera, una arista está caracterizada por dos pares (i, b) y (i', b') de tales números, que representan a las dos semiaristas de sus extremos.

Definición 1.2.1 *Sea* $\mathcal{D} : G \to S$ *un diagrama en una superficie* S *con sus vértices numerados* $\{1, \ldots, n\}$ *de manera arbitraria. Sea* m_i, $i = 1, \ldots, n$ *la valencia del vértice* i. *Supongamos que para cada vértice* i *se ha elegido una orientación local de un entorno y se han numerado las semiaristas de* i *con los números* $\{1, \ldots, m_i\}$ *siguiendo la orientación local elegida y empezando en una de ellas elegida de manera arbitraria.*

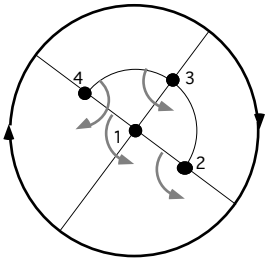
Diremos que una arista está bien orientada *si transportando a su través la orientación elegida en uno de sus extremos coincide con la elegida en el otro extremo. Diremos que está* mal orientada *en caso contrario.*



$([1_1, 4_1], [1_2, 5_1], [1_3, 2_1], [1_4, 1_5], [2_2, 3_1], [3_2, 4_4], [3_3, 4_4], [3_5, 4_3])$



$([1_1/4_2], [1_2, 2_2], [1_3, 2_1], [1_4, 3_2], [2_3, 3_3], [3_1/4_1][3_4/4_3])$



$([1_1/4_2], [1_2/3_4], [1_3, 2_1], [1_4, 3_2], [2_2, 4_3], [2_3, 3_3], [3_1/4_1])$

Figura 1.5: Algunos diagramas y sus correspondientes listas de aristas.

Llamaremos lista de aristas *asociada a $\mathcal{D}$ para las elecciones efectuadas a la lista formada por las aristas de $\mathcal{D}$ representadas cada una de ellas por los dos pares de números asociados a sus semiaristas y la información de si se trata de una arista bien orientada o mal orientada.*

Para representar una arista de la lista de forma compacta normalmente escribiremos el número que indica el orden de la semiarista como subíndice del que indica el vértice. Además, distinguiremos a las aristas "bien orientadas" de las "mal orientadas", según nuestra definición, escribiendo las primeras en la forma $[i_b, i'_{b'}]$ y las segundas en la forma $[i_b/i'_{b'}]$.

En la Figura 1.5 se dan tres diagramas (en un toro, una esfera y un plano proyectivo, respectivamente) con sus correspondientes listas de aristas. La flecha gris alrededor de cada vértice indica la orientación y semiarista inicial (a la que se asigna el número '1' en la lista) elegidas en cada vértice. En el caso de una superficie orientable podemos elegir todas las orientaciones locales compatibles con una cierta orientación global, con lo cual todas las aristas quedan automáticamente

'bien orientadas'. Ésto es lo que ocurre en el primer diagrama de la Figura, pero no ocurre en el segundo.

Dado un cierto diagrama $\mathcal{D}$, el número de posibles listas asociadas es, a lo sumo, igual al número de posibles formas de elegir la numeración de los vértices, la orientación de los vértices y la numeración de las semiaristas. Este número es $n!2^n \prod_{i=1}^{n} m_i$, donde n es el número de vértices y m_i la valencia del vértice i. En algunos casos dos elecciones diferentes producirán la misma lista, lo cual está relacionado con la existencia de simetría en el diagrama. Por ejemplo, los dos últimos diagramas de la Figura 1.5 poseen una simetría que intercambia el vértice 1 por el 3 y el 2 por el 4. Aún así, seguirá habiendo un número elevado (exponencial) de posibles listas de aristas asociadas al diagrama. En el próximo apartado vamos a ver cómo definir y cómo obtener una de ellas en particular a la que llamaremos la *lista canónica* asociada al diagrama.

1.2.2 Obtención de una lista canónica

Aunque el orden en que las aristas aparecen en una lista de aristas y el orden en que las dos semiaristas de una arista aparecen en ella no está fijado por nuestra definición (es decir, se consideran como listas no ordenadas) normalmente escribiremos ambas cosas ordenadas, entendiendo por ésto que $1_1 < 1_2 < \ldots 1_{m_1} < 2_1 < 2_2 < \ldots < n_1 < \ldots n_{m_n}$ para las semiaristas y que dadas dos aristas a_1 y a_2, a_1 es menor que a_2 si y sólo si la primera semiarista de a_1 es menor que la primera de a_2. Obsérvese que las listas de la Figura 1.5 están ordenadas de esta manera.

Ésto nos sirve también para comparar aristas que procedan de diferentes listas, asociadas a un mismo diagrama o no. Dadas dos aristas cualesquiera, diremos que es menor la que tiene menor su primera semiarista o, si las primeras semiaristas coinciden, la que tiene menor la segunda; si ambas semiaristas coinciden y las dos aristas se distinguen sólo porque una está bien orientada y la segunda no, entonces tomamos como convenio que la que está bien orientada es menor que la que está mal orientada.

Por último, podemos también comparar listas. Dadas dos listas diferentes L_1 y L_2 ordenadas según el convenio descrito, diremos que la lista más pequeña es la que tiene la arista más pequeña en la primera posición en la que las dos listas no coinciden. Si sucede que las listas son de longitudes diferentes $l_1 < l_2$ y las l_1 aristas de L_1 coinciden con las primeras l_1 aristas de L_2, diremos que L_1 es menor que L_2. En particular, ésto se aplica a la lista vacía, que resulta ser menor que cualquier otra.

El orden que acabamos de definir sobre el conjunto de listas, al que llamaremos *lexicográfico*, es un orden total. Puesto que el número de posibles listas asociadas a un diagrama es finito, tiene sentido definir la lista canónica de la siguiente manera.

Definición 1.2.2 *Sea $\mathcal{D}$ un diagrama y sean $L_1, \ldots, L_N$ todas las posibles listas de aristas asociadas a $\mathcal{D}$. Llamaremos* lista de aristas canónica *asociada a $\mathcal{D}$ a la menor de todas ellas, con el orden lexicográfico arriba definido.*

Por supuesto, hay formas mejores de calcular la lista canónica de un diagrama $\mathcal{D}$ que construir todas las posible listas de $\mathcal{D}$. El método que nosotros vamos a usar se basa en dos pequeñas observaciones.

Lema 1.2.3

(i) Sea L una lista de aristas ordenada lexicográficamente. Entonces, la primera semiarista de la arista i-ésima de L es la más pequeña, lexicográficamente, de las semiaristas de las aristas en posición j-ésima, $j \geq i$.

(ii) Sea L_C la lista de aristas canónica de un cierto diagrama $\mathcal{D}$, ordenada lexicográficamente. Entonces, la segunda semiarista de la arista i-ésima de L_C o bien es una semiarista incidente a un cierto vértice que ya ha aparecido en alguna arista anterior a la i-ésima o bien es la semiarista número 1 del vértice con el número más pequeño de los que aún no han aparecido. En este caso, además, dicha arista estará bien orientada en la lista. □

Aún no sabemos cómo averiguar cuál es la semiarista que va a estar nombrada 1_1 en la lista canónica, pero sí tenemos ya una manera rápida de calcular el resto de la lista canónica para un diagrama conexo, si sabemos cuál es dicha arista y cuál es la orientación en dicho vértice. A saber, si conocemos las $i-1$ primeras aristas de la lista y la primera semiarista de la i-ésima, miraremos en el diagrama (o en una lista asociada que conozcamos) cuál es el segundo extremo de dicha arista. En el caso de que alguna de las semiaristas que inciden en el mismo vértice que ella ya esté en la lista, ésto proporciona una manera única de nombrar a dicha semiarista y nos dice si la arista está bien orientada o no. En caso contrario, la propiedad (ii) del Lema anterior nos dice cómo nombrar a dicha segunda semiarista. También nos dice que la arista va a estar bien orientada y nos dice cuál va a ser, por tanto, la orientación de este nuevo vértice. Por otra parte, si conocemos las i primeras aristas, la primera semiarista de la $i+1$-ésima viene dada por la propiedad (i) de este mismo Lema.

Aclaremos este proceso con un ejemplo, que es el segundo diagrama de la Figura 1.5. El proceso se puede llevar a cabo directamente a partir de la lista que conocemos (que es $L = ([1_1/4_2], [1_2, 2_2], [1_3, 2_1], [1_4, 3_2], [2_3, 3_3], [3_1/4_1][3_4/4_3])$) pero probablemente mirar el diagrama ayude al lector a seguir el proceso de una forma geométrica que, en realidad, no es otra cosa que una *breadth first search* (búsqueda primero en anchura) en el grafo del diagrama. Supongamos que la semiarista 1_1 no cambia de nombre, y que el vértice 1 no cambia de orientación. Lo siguiente es el proceso de obtención de la lista canónica (módulo la suposición que hemos hecho) detallada paso a paso. En cada paso, la lista *Dict* es un "diccionario" que contiene toda la información que ya conocemos sobre cómo se traducen los nombres de la lista L a la lista canónica. Cada término del diccionario tiene la misma estructura que una arista de la lista, pero su lectura es diferente. Aquí $[i_b, i'_{b'}]$ significa que la semiarista i_b de la antigua lista se llama $i'_{b'}$ en la nueva lista, y que la orientación en torno al vértice correspondiente no cambia de una lista a la otra. $[i_b/i'_{b'}]$ significa que la semiarista i_b de la antigua lista se llama $i'_{b'}$ en la nueva lista, y que la orientación en torno al vértice correspondiente cambia. Nótese que en el diccionario sólo necesitamos guardar la "traducción" correspondiente a una de las semiaristas de cada vértice. Las demás se deducen de ella. Al principio sólo conocemos $Dict = ([1_1, 1_1])$ y $L_C = ([1_1, \ldots)$. A partir de aquí iremos averiguando el resto:

$$L_C = ([1_1, \ldots) \quad Dict = ([1_1, 1_1])$$

$$L_C = ([1_1, 2_1], [1_2, \ldots) \quad Dict = ([1_1, 1_1], [4_2/2_1]$$

$$L_C = ([1_1, 2_1], [1_2, 3_1], [1_3 \ldots) \quad Dict = ([1_1, 1_1], [4_2/2_1], [3_4, 3_1])$$

$$L_C = ([1_1, 2_1], [1_2, 3_1], [1_3, 4_1], [1_4, \ldots) \quad Dict = ([1_1, 1_1], [4_2/2_1], [3_4, 3_1], [2_1, 4_1])$$

$$L_C = ([1_1, 2_1], [1_2, 3_1], [1_3, 4_1], [1_4, 3_3], [2_2, \ldots)$$
$$Dict = ([1_1, 1_1], [4_2/2_1], [3_4, 3_1], [2_1, 4_1])$$

$$L_C = ([1_1, 2_1], [1_2, 3_1], [1_3, 4_1], [1_4, 3_3], [2_2, 3_2], [2_3, \ldots)$$
$$Dict = ([1_1, 1_1], [4_2/2_1], [3_4, 3_1], [2_1, 4_1])$$

$$L_C = ([1_1, 2_1], [1_2, 3_1], [1_3, 4_1], [1_4, 3_3], [2_2, 3_2], [2_3, 4_2], [3_4, \ldots)$$
$$Dict = ([1_1, 1_1], [4_2/2_1], [3_4, 3_1], [2_1, 4_1])$$

$$L_C = ([1_1, 2_1], [1_2, 3_1], [1_3, 4_1], [1_4, 3_3], [2_2, 3_2], [2_3, 4_2], [3_4, 4_3])$$
$$Dict = ([1_1, 1_1], [4_2/2_1], [3_4, 3_1], [2_1, 4_1])$$

Para un diagrama conexo, el proceso puede ser continuado siempre de manera única hasta que hemos traducido todas las aristas. Si, en cambio, el diagrama no fuera conexo, el proceso recorrería todas las aristas de una componente conexa y, después, no sabría cómo continuar. Para listas no conexas, en un primer paso separaremos la lista en componentes conexas (Proposición 1.2.4).

El algoritmo para calcular la lista canónica consiste en realizar este proceso comenzando por cada una de las semiaristas del diagrama y con las dos posibles orientaciones para cada una. Se obtienen así $4l$ posibles listas canónicas de la misma longitud l que la lista original. La verdadera lista canónica se obtendrá comparando estas $4l$ entre sí. Nótese que l es el parámetro natural para medir la complejidad del algoritmo y coincide con el número de aristas del diagrama.

Proposición 1.2.4 *Sea L una lista de aristas de longitud l asociada a un diagrama $\mathcal{D}$. Existe un algoritmo que, con* input *L, calcula la lista canónica L_C asociada a $\mathcal{D}$ en tiempo* $\mathrm{O}(l^2)$.

Demostración: Supongamos, de momento, que el diagrama es conexo. Ésto se refleja en la lista en el hecho de que no puede ser partida en dos sublistas que contengan subconjuntos disjuntos de vértices.

Para bajar el tiempo asintótico de ejecución del algoritmo utilizaremos punteros que conecten unas aristas de la lista con otras. Supondremos que el tiempo que toma atravesar un puntero (i.e., el acceso a memoria) es constante y que cada operación aritmética que interviene también se realiza en tiempo constante. Nótese

que las operaciones que intervienen son aritmética módulo las distintas valencias de los vértices y comparaciones de números enteros acotados por l.

El primer paso del algoritmo será crear una "matriz de punteros" a las semiaristas de la lista, de forma que encontrar la arista que contiene a una cierta semiarista i_b se pueda hacer en tiempo constante. Ésto se puede hacer en tiempo cuadrático.

Después, para cada semiarista y cada orientación de su vértice, tomaremos dicha semiarista como la 1_1 y reconstruiremos la lista canónica que resultaría si esa elección fuera la acertada. Hay por tanto $4l$ reconstrucciones a hacer.

En la reconstrucción llevaremos siempre la cuenta de cuál es la semiarista mayor que ha aparecido como primera parte de una arista de la lista y cuál es el vértice mayor que ha aparecido en la lista. Además, guardaremos dos copias del diccionario (una ordenada según nombres antiguos y otra según nombres nuevos) de forma que cada consulta para traducir un nombre antiguo a nuevo o uno nuevo a antiguo se hace en tiempo constante. Por último, llevaremos también una matriz de punteros, análoga a la de la lista antigua, para las semiaristas de la lista nueva que ya conozcamos.

En estas condiciones, para añadir la segunda semiarista de una arista necesitamos: traducir el nombre nuevo a antiguo, buscar la arista de L que contiene dicho nombre antiguo, tomar su otro extremo y traducir el nombre antiguo que aparece a nuevo. Si el nombre antiguo no aparece en nuestro diccionario, le daremos el nombre que corresponde por el Lema y lo añadiremos al diccionario. Todo ello se hace en tiempo constante. Por tanto, el tiempo total invertido en esta parte es $\mathrm{O}(l)$.

Por otra parte, para añadir la primera semiarista de una nueva arista a la lista, consideraremos la siguiente a la mayor que ha aparecido como primera semiarista de la lista, y miraremos si también ha aparecido. Si es así la saltaremos y continuaremos con la siguiente. Si no es así la tomaremos como buena. Aunque el añadido de cada una no es en tiempo constante (porque puede haber varias a desechar antes de encontrar la que nos hace falta) en total habremos probado una vez con cada una de las $2l$ aristas. Por tanto, el tiempo total invertido en esta parte es también $\mathrm{O}(l)$.

Como el proceso de reconstrucción a partir de una semiarista inicial es lineal, el proceso total de obtención de las $4l$ listas candidatas es cuadrático. Una vez obtenidas todas ellas, necesitamos averiguar cuál es la más pequeña, que será la canónica. Ésto se puede hacer con $4l - 1$ comparaciones y cada una de ellas lleva $\mathrm{O}(l)$. Por tanto, la lista canónica se obtiene en tiempo $\mathrm{O}(l^2)$.

Para un diagrama no conexo tenemos una pequeña complicación añadida, pero ésta no afecta al tiempo asintótico total:

Como primer paso dividiremos la lista L en k listas $L_1, \ldots, L_k$ conexas de longitudes $l_1, \ldots l_k$, $\sum l_i = l$, del mismo modo que se separa un grafo en componentes conexas. Cada sublista se corresponde con una componente conexa del diagrama pero sólo aquéllas componentes conexas del diagrama que poseen aristas (es decir, las que no son puntos aislados) se corresponden con alguna sublista.

Después traduciremos cada sublista a su forma canónica por el algoritmo anterior, en tiempo $\mathrm{O}(l_i^2 + \cdots + l_k^2) \leq \mathrm{O}(l^2)$. Las diferentes sublistas canónicas tendrán todas sus vértices numerados comenzando en '1' (es decir, vértices de diferentes componentes pueden recibir el mismo nombre) pero ésto no producirá confusión porque, en este punto, las sublistas se mantienen separadas. El conjunto de estas sublistas canónicas puede ser ordenado lexicográficamente en tiempo $\mathrm{O}(l^2)$ de la

siguiente manera: comparar lexicográficamente una lista L_i de longitud l_i con todas las demás lleva tiempo, a lo sumo, $l \times l_i$. Tomando una de ellas al azar dividimos a todas las demás en tres grupos formados por las que son mayores, las que son menores y las que son iguales a L_i. A partir de aquí sólo necesitamos comparar entre sí las de los dos primeros subgrupos, y L_i no volverá a intervenir.

Una vez ordenadas las sublistas canónicas, para ser consecuentes con nuestra definición de lista asociada, deberemos renumerar los vértices para que no haya dos con el mismo nombre antes de yuxtaponer las listas. Esta renumeración se hará empezando por la lista más pequeña y se puede hacer en tiempo lineal. □

Aunque no se nos ocurre ningún método para rebajar la complejidad asintótica del caso peor del algoritmo, sí hay trucos que pueden usarse para reducir el tiempo de ejecución en la mayoría de los casos. Estos trucos se basan en restringir el número de semiaristas iniciales en las que hay que buscar. Por ejemplo, en diagramas que tengan bucles, la primera arista del código canónico va a ser siempre uno de ellos, lo cual reduce considerablemente la búsqueda. También se puede modificar la definición de código canónico en casos particulares. Por ejemplo, para diagramas en el plano afín (es decir, para diagramas en la esfera con un punto distinguido) podemos tomar por convenio que el código canónico tenga por primera semiarista a una de las que inciden en él (esto es, las ramas al infinito del diagrama en el plano) o, en su defecto, alguna de las que forman parte del contorno del mismo.

1.2.3 Caracterización de un entorno regular

Vamos a ver ahora la relación existente entre las listas de aristas definidas en el apartado anterior y los entornos regulares de un diagrama, la cual viene dada por el siguiente enunciado:

Teorema 1.2.5 *Sean $\mathcal{D}_1 : G_1 \to S_1$ y $\mathcal{D}_2 : G_2 \to S_2$ dos diagramas en sendas superficies compactas. Entonces son equivalentes:*

(i) Existen sendos entornos regulares N_1 y N_2 de los diagramas $\mathcal{D}_1$ y $\mathcal{D}_2$ y un homeomorfismo $h : N_1 \to N_2$ que envía biyectivamente $\mathcal{D}_1(G_1)$ sobre $\mathcal{D}_2(G_2)$ y los vértices de D_1 sobre vértices de D_2.

(ii) Cualquier lista de aristas L asociada a $\mathcal{D}_1$ es también una de las posibles listas de aristas asociada a $\mathcal{D}_2$ y $\mathcal{D}_1$ y $\mathcal{D}_2$ tienen el mismo número de vértices aislados.

(iii) Las listas de aristas canónicas asociadas a $\mathcal{D}_1$ y a $\mathcal{D}_2$ coinciden y ambos tienen el mismo número de vértices aislados.

(iv) Existe una lista de aristas L asociada a $\mathcal{D}_1$ que es también una de las posibles listas de aristas asociada a $\mathcal{D}_2$ y $\mathcal{D}_1$ y $\mathcal{D}_2$ tienen el mismo número de vértices aislados.

Demostración: Las implicaciones $(ii) \Rightarrow (iii) \Rightarrow (iv)$ son obvias. También es fácil $(i) \Rightarrow (ii)$ sin más que utilizar el homeomorfismo para trasladar las elecciones de numeración y orientaciones establecidas en el primer diagrama al segundo. Sólo nos queda ver la implicación $(iv) \Rightarrow (i)$. Reduzcamos en primer lugar la afirmación al caso de diagramas sin vértices aislados. Por la Proposición 1.1.9 cada componente

conexa del entorno regular de un diagrama que contenga un vértice aislado es un disco y no contiene más partes del diagrama. Por tanto, dados dos diagramas con el mismo número de vértices aislados, todo homeomorfismo como el de (i) para las componentes de los entornos regulares que no contienen puntos aislados se puede extender a las componentes que tengan puntos aisados. Ésto implica que si la implicación es cierta para diagramas sin vértices aislados, la implicación será también cierta para diagramas con vértices aislados.

Supongamos ahora que $\mathcal{D}_1$ y $\mathcal{D}_2$ son diagramas sin vértices aislados (ésto es, que todo vértice posee al menos una semiarista que incide en él). Vamos a construir, partiendo únicamente de una lista L asociada a $\mathcal{D}_1$ una cierta superficie con borde N_L y un diagrama $\mathcal{D}_L : G_L \to N_L$. Después, construiremos un entorno regular N_1 de $\mathcal{D}_1$ y un homeomorfismo $h_1 : N_1 \to N_L$ que envía biyectivamente $\mathcal{D}_1(G_1)$ sobre $\mathcal{D}_L(G_L)$ y los vértices de $\mathcal{D}_1$ sobre vértices de $\mathcal{D}_L$. De manera análoga se podrá construir un entorno regular N_2 de $\mathcal{D}_2$ y un homeomorfismo $h_2 : N_2 \to N_L$ en las mismas condiciones, y tomando $h = h_2^{-1} \circ h_1$ habremos terminado la demostración.

Sea entonces L una lista asociada a $\mathcal{D}_1$. El diagrama $\mathcal{D}_L$ se construye de la siguiente manera. A partir de la lista L conocemos el número de vértices n que tiene el diagrama $\mathcal{D}_1$ y las valencias $m_1, \ldots, m_n$ de cada uno de ellos. Tomemos n círculos $C_1, \ldots, C_n$ y en cada uno de ellos tantos radios $r_1^i, \ldots, r_{m_i}^i$ como la valencia del vértice i. Asignemos una cierta orientación a cada círculo y supongamos que los radios están numerados siguiendo dicha orientación. Llamemos I_b^i a un pequeño arco (cerrado) del borde del círculo C_i con el extremo del radio r_b^i en su interior.

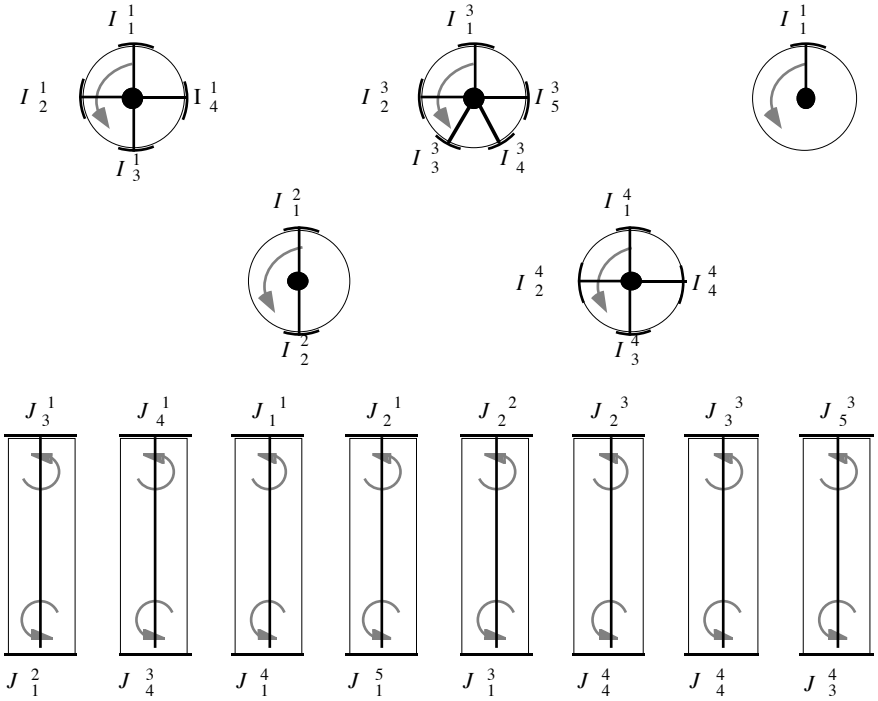


Figura 1.6: Reconstrucción de un entorno regular, a partir de una lista.

Del mismo modo, por cada arista $a = [i_b, i'_{b'}]$ o $a = [i_b / i'_{b'}]$ que aparezca en la lista construyamos un rectángulo R_a y un segmento s_a que una los dos puntos medios de dos lados opuestos. Llamemos J_b^i y $J_{b'}^{i'}$ a dichos lados de R_a. En cada R_a,

orientemos sendos entornos de los correspondientes J_b^i y $J_{b'}^{i'}$ de forma compatible si la arista a estaba bien orientada en la lista y de forma incompatible si estaba mal orientada. La Figura 1.6 muestra esta construcción para la lista del primer diagrama de la Figura 1.5 (en ella todas las aristas estaban bien orientadas).

La construcción del entorno regular se termina considerando la unión de todos los círculos C_i y rectángulos R_a, en la cual se identifica cada arco I_b^i de un círculo con el correspondiente lado J_b^i de un rectángulo (haciendo la identificación de forma que la orientación del círculo concuerde con la orientación en un entorno del lado del rectángulo. La superficie con borde resultante de las identificaciones será la N_L que buscamos.

Los vértices del diagrama serán los centros de los círculos C_i y cada arista será la unión del segmento s_a en uno de los rectángulos con los dos radios r_b^i y $r_{b'}^{i'}$ correspondientes a las semiaristas i_b y $i'_{b'}$ de la arista $a = [i_b, i'_{b'}]$ (o $a = [i_b/i'_{b'}]$). Ésto define un diagrama $\mathcal{D}_L$ en N_L.

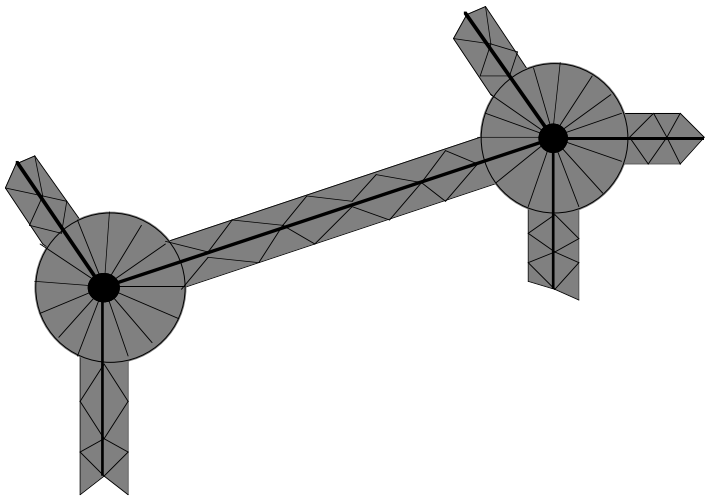


Figura 1.7: Entorno regular.

La existencia del homeomorfismo $h_1 : N_1 \to N_L$ se demuestra de la siguiente manera: en primer lugar es fácil demostrar que se puede encontrar un entorno regular N_1 de $\mathcal{D}_1(G_1)$ que sea localmente como el que se muestra en la Figura 1.7. Basta para ello triangular de forma adecuada un cierto entorno de $\mathcal{D}(G)$. Una vez hecho ésto, podemos enviar homeomórficamente cada círculo en torno de un vértice i de $\mathcal{D}_1$ sobre el correspondiente círculo C_i de N_L conservando la orientción, enviando la semiarista de $\mathcal{D}_1$ que ha sido nombrada i_b sobre el radio r_b^i y enviando el arco de círculo correspondiente sobre el arco I_b^i. Luego, para cada arista a, extandamos el homeomorfismo enviando cada banda en torno a una arista de $\mathcal{D}$ sobre el rectángulo R_a de la arista de $\mathcal{D}_L$ que lleva el mismo nombre. El hecho de que las aristas de $\mathcal{D}_L$ se correspondan con aristas de $\mathcal{D}_1$ con la misma representación en la lista L nos asegura que el homeomorfismo se puede extender a todo N_1. □

1.2.4 Diagramas celulares

El Teorema 1.2.5 tiene algunas consecuencias inmediatas de bastante interés para los diagramas cuyas caras sean celdas (i.e. cuyas caras regulares sean discos, a la

luz de la Proposición 1.1.9). En su demostración hemos construido, a partir de una lista L asociada a un cierto diagrama $\mathcal{D}_1$, otro diagrama $\mathcal{D}_L$ en una superficie N_L (con borde), de forma que L es también lista asociada al diagrama $\mathcal{D}_L$. Como nos hemos propuesto trabajar siempre en superficies sin borde, peguemos un disco en cada componente del borde de N_L (que es una circunferencia) para obtener una cierta superficie S_L compacta y sin borde. En estas condiciones $\mathcal{D}_L : G_L \to S_L$ se convierte en un diagrama en la superficie S_L con entorno regular N_L. Además, las caras regulares del diagrama $\mathcal{D}_L$ son los discos que hemos pegado a N_L y, por tanto, el diagrama es celular. Estas consideraciones llevan de forma inmediata al siguiente enunciado, que podríamos parafrasear diciendo que *toda lista de aristas es realizable como la lista de aristas de un diagrama celular.*

Corolario 1.2.6 *Sean $n \in \mathbb{N}$ y $m_1, \ldots, m_n \in \mathbb{N}$ números naturales. Sea L una lista cuyos elementos son pares $[i_b, i'_{b'}]$ o $[i_b / i'_{b'}]$ de símbolos de la forma i_b, $i \in \{1, \ldots, n\}$, $b \in \{1, \ldots, m_i\}$ donde cada símbolo aparece una y sólo una vez. Entonces, existe un cierto diagrama celular $\mathcal{D}_L : G_L \in S_L$ en una superficie compacta S_L que tiene a L como una de sus posibles listas asociadas.* □

Además, el Teorema 1.2.5 implica también unicidad del diagrama celular asociado a una cierta lista. Ésto es así porque, para diagramas celulares, el homeomorfismo h de N_1 en N_2 se puede extender a las caras regulares por ser éstas discos. Este hecho es especialmente significativo para diagramas en la esfera porque los diagramas conexos en la esfera son siempre celulares.

Corolario 1.2.7 *Sean $\mathcal{D}_1 : G_1 \to S_2$ y $\mathcal{D}_2 : G_2 \to S_2$ dos diagramas celulares en sendas superficies compactas S_1 y S_2. Entonces, $\mathcal{D}_1$ y $\mathcal{D}_2$ son combinatoriamente equivalentes si y sólo si están en las cuatro condiciones (equivalentes) del Teorema 1.2.5.* □

Por último, veamos que existe una manera sencilla de averiguar cuál es la superficie S_L en la que se realiza el diagrama celular $\mathcal{D}_L$ determinado por una cierta lista L. Como es bien sabido, una superficie compacta queda determinada si conocemos, para cada una de sus componentes conexas, su orientabilidad y su característica de Euler. Dada una lista L, es fácil separar L en sublistas $L_1, \ldots, L_k$ que representen las componentes conexas del diagrama.

Para cada sublista, la componente S_k de la superficie será orientable si y sólo si lo es el entorno regular N_k y éste lo es si existe una cierta aplicación de los vértices de la lista en el conjunto $\{-1, +1\}$ tal que vértices conectados por una arista bien orientada tienen la misma imagen y vértices conectados por una arista mal orientada tienen imágenes diferentes. Averiguar la existencia o no de una función en esas condiciones es fácil: asígnese el valor -1 a uno de los vértices y propáguese la función al resto de ellos de forma coherente con un cierto subconjunto de aristas que sea un árbol maximal del pseudografo. Si la función obtenida es también coherente con el resto de las aristas la respuesta es afirmativa y si no, será inútil que sigamos buscando la función: habremos encontrado un circuito en el pseudo-grafo en el cual la orientación cambia un número impar de veces.

En cuanto a la característica de Euler, para una descomposición celular de una superficie compacta (y nuestro diagrama lo es) la característica de Euler es igual a

$F - A + V$, donde F A y V son los números de caras, aristas y vértices, respectivamente. Los números de vértices y aristas en la componente S_k son trivialmente obtenibles de la lista L_k. Para obtener el número de caras nos basaremos en la técnica de seguir las componentes conexas del borde del entorno regular. Para un diagrama celular, las componentes conexas del borde del entorno regular están en correspondencia biyectiva con las caras).

Supongamos que queremos seguir el borde de la cara que está *a la derecha* de la semiarista i_b. Nótese que, puesto que el vértice i está dotado de una orientación local y la semiarista i_b puede considerarse como un segmento dirigido con origen en i, tiene sentido hablar de la cara a la derecha (o a la izquierda) de i_b. Si la arista que contiene a i_b en la lista es de la forma $i'_{b'}$, entonces, la cara a la derecha de i_b es la misma que la cara a la izquierda de $i'_{b'}$. Si la arista es $[i_b/i'_{b'}]$, entonces la cara es la que está a la izquierda de $i'_{b'}$. Por otra parte, la arista a la derecha de i_b es la misma que la que está a la izquierda de i_{b-1} (donde $b-1$ se considera módulo m_i). Siguiendo estas dos reglas, podemos recorrer las caras del diagrama celular en orden circular. Cuando, recorriéndolas de esta forma, hayamos pasado dos veces por cada semiarista de L_k (una a la derecha y otra a la izquierda) habremos recorrido todas la caras.

La realizabilidad de toda lista como diagrama celular y la unicidad de ésta realización son de utilidad a la hora de construir un catálogo de tipos combinatorios de diagramas celulares. A saber, si queremos conocer todos los tipos de tales diagramas que existen con a lo más un cierto número n de vértices, con valencias a lo más $m_1, \ldots, m_n$, bastará con construir todas las posibles listas en esos símbolos y estudiar el diagrama producido por cada una de ellas. Por supuesto, es necesario poseer una forma de decidir si dos listas diferentes producen diagrams celulares equivalentes o no. Ésto puede hacerse, gracias al Teorema 1.2.5, traduciendo ambas listas a su forma canónica como en la Sección 1.2.2.

1.3 Codificación completa del diagrama. El grafo de componentes conexas

1.3.1 El grafo de componentes conexas

Cuando tratemos con diagramas no celulares la información recogida en la lista de aristas no será suficiente para caracterizarlos módulo equivalencia. Una demostración trivial de ésto es que en la Proposición 1.2.5 vimos que cualquier lista de aristas puede ser realizada por un diagrama celular. Es decir, toda lista de aristas representa en realidad a varias clases de equivalencia de diagramas diferentes, de las cuales sólo una es celular.

Es claro que para codificar diagramas no celulares (es decir, no conexos y/o con caras no simplemente conexas) la información que debemos añadir a la lista de aristas es, por una parte, la información topológica de cada una de las caras y, por otra, la información de cómo cada cara está unida al diagrama. Desde otro punto de vista, como la lista de aristas L codifica al entorno regular N del diagrama, lo que nos queda por codificar es el complemento regular M del mismo y la forma en que N y M están pegados uno a otro por su borde, que es común.

La codificación de M es sencilla, puesto que cada componente conexa de M

(cada cara regular del diagrama) es una superficie conexa con borde cuya topología queda caracterizada conociendo su orientabilidad, género y número de componentes conexas de su borde. Para codificar el pegado entre el entorno regular y el complemento regular utilizaremos un grafo bipartito cuyas aristas estarán en correspondencia biunívoca con las componentes de $N \cap M$ (a las que llamaremos *ciclos de borde* del diagrama) y cuyos vértices con las componentes conexas de M y N respectivamente.

Llamaremos a este grafo el *grafo de componentes conexas* $\mathcal{G}$ del diagrama. En realidad se trata de un *multigrafo*, puesto que una misma cara y componente conexa pueden estar unidas por más de un ciclo de borde. Para evitar confusión con las aristas y vértices del diagrama $\mathcal{D}$, evitaremos el uso de las palabras arista y vértice al referirnos a los elementos del grafo de componentos conexas. Más bien, llamaremos *nodos de cara* y *nodos de componente* a sus vértices (que representan, respectivamente, a las caras y componentes conexas del diagrama) y *elementos de borde* a sus aristas.

Toda la información topológico/combinatoria del diagrama irá "prendida" al grafo de la siguiente manera.

- La lista de aristas (canónica o no) asociada al diagrama será separada en componentes conexas, como se hizo en la demostración de 1.3.7. Cada sublista se prenderá a su correspondiente nodo de componente. Las componentes que sean puntos aislados llevarán una "lista vacía". Para las demás, podemos suponer que los vértices de cada componente conexa han sido renumerados comenzando en 1.

- Los tres invariantes de cada cara regular irán prendidos al correspondiente nodo de cara.

- En cuanto a los elementos de borde, que unen una cierta cara f a una componente c, llevarán prendida una de las semiaristas que tenga a la cara f a su derecha. Recordemos que las semiaristas están dirigidas y los vértices tienen una orientación local, dada por la lista asociada. Por tanto, tiene sentido decir que una cara está a la derecha o a la izquierda de una semiarista.

Para las caras orientables con más de un elemento de borde necesitamos una información adicional: saber si las orientaciones inducidas en sus elementos de borde por las semiaristas que los señalan son compatibles o no. De manera más explícita: para cada cara orientable elegiremos una de sus dos posibles orientaciones globales. Ésto induce una orientación a la derecha de la semiarista elegida para señalar a cada ciclo de borde, la cual puede coincidir o no con la orientación local del vértice en el que incide la semiarista. Si coincide, diremos que el ciclo de borde está *bien orientado*; si no, que está *mal orientado*. Nótese que una semiarista puede tener el mismo ciclo de borde a sus dos lados y que la orientación dada a la cara sea contradictoria en uno y otro (e.g. un diagrama cuya imagen es una recta en el plano proyectivo). Puesto que estamos pensando en la cara como "a la derecha de la semiarista" las orientaciones han de compararse también por la derecha.

Cada elemento de borde del grafo de componentes conexas que sea adyacente a una cara orientable contendrá la información de si el ciclo de borde correspondiente está bien orientado o mal orientado. Para las caras no orientables ésta información no tiene sentido, pero tampoco es necesaria dado que:

Lema 1.3.1 *Sean S_1 y S_2 dos superficies con borde, conexas y homeomorfas. Sea $h : \partial S_1 \to \partial S_2$ un homeomorfismo entre sus bordes. Entonces,*

(i) Si las superficies no son orientables, el homeomorfismo se puede siempre extender a toda la superficie.

(ii) Si las superficies son orientables, el homeomorfismo se puede extender si y solamente si las orientaciones inducidas por h en ∂S_2 a partir de una orientación global de S_1 son compatibles con una orientación global de S_2. □

Veamos ahora que el grafo de componentes conexas, con esta información añadida, caracteriza al diagrama módulo equivalencia, mediante la siguiente Definición y Teorema, parecidos en su forma a la Definición 1.2.1 y el Teorema 1.2.5.

Definición 1.3.2 *Sea $\mathcal{D} : G \to S$ un diagrama en una superficie S. Supongamos que: para cada componente conexa C de $\mathcal{D}$, se ha construido una lista de aristas asociada a ella; para cada cara orientabe F de $\mathcal{D}$, se ha elegido una orientación de manera arbitraria; para cada ciclo de borde de $\mathcal{D}$, se ha elegido una de las semiaristas que lo tienen a la derecha.*

Diremos que un ciclo de borde de una cara orientable está bien orientado *si las orientaciones en él inducidas por la semiarista elegida para él y la cara que lo contiene coinciden.*

Llamaremos código del diagrama *al grafo de componentes conexas del mismo, arriba descrito, dando como información: para cada nodo de componente, su lista asociada; para cada nodo de cara, sus tres invariantes; para cada elemento de borde, la semiarista elegida para el ciclo de borde y el dato de si el ciclo está bien orientado o mal orientado.*

Teorema 1.3.3 *Sean $\mathcal{D}_1 : G_1 \to S_1$ y $\mathcal{D}_2 : G_2 \to S_2$ dos diagramas en sendas superficies compactas. Entonces son equivalentes:*

(i) Los diagramas son equivalentes.

(ii) Cualquier código C asociado a $\mathcal{D}_1$ es también uno de los posibles códigos asociados a $\mathcal{D}_2$.

(iii) Existe un código C asociado a $\mathcal{D}_1$ que es también uno de los posibles códigos asociados a $\mathcal{D}_2$.

Demostración: $(i) \Rightarrow (ii) \Rightarrow (iii)$ son triviales. Para $(iii) \Rightarrow (i)$, deberemos ser capaces de construir un homeomorfismo de S_1 en S_2 que envíe un diagrama sobre el otro, a partir de la información contenida en el código. El Teorema 1.2.5 nos dice cómo construir un homeomorfismo entre sendos entornos regulares N_1 y N_2 de los diagramas, a partir de las listas asociadas. La estructura del grafo nos asegura que dicho homeomorfismo lleva el borde de cada cara regular sobre el borde de otra cara regular homeomorfa a ella y que el homeomorfismo de los entornos regulares se extiende al interior de las caras regulares, por el Lema 1.3.1. □

Casi todo lo dicho en la Sección 1.2.4 sobre la relación entre las listas de aristas y los diagramas celulares tiene una traducción más o menos inmediata a la relación entre diagramas en general y sus códigos asociados.

Corolario 1.3.4 *Sea $\mathcal{G}$ un multigrafo bipartito a cuyas dos clases de nodos llamaremos, respectivamente, nodos de cara y nodos de componente y a cuyas aristas llamaremos elementos de borde.*

Supongamos que: a cada nodo de componente c se le asocia una lista de aristas arbitraria (quizá vacía) como las del Corolario 1.3.4, cuyo diagrama celular asociado tiene tantas caras como la valencia de c; a cada nodo de cara f se le asocian los tres invariantes de una superficie compacta con borde, con tantas componentes de borde como la valencia de f; a cada elemento de borde que une un nodo cara f con un nodo componente c se le asocia una semiarista de los ciclos de borde de la lista de f, asociando a elementos de borde distintos semiaristas que tienen a su derecha ciclos de borde distintos; por último, a los ciclos de borde incidentes sobre nodos de caras orientables, se les asigna un número 1 *ó* 0 *que se leerá como "el ciclo de borde está bien orientado" o "el ciclo de borde está mal orientado".*

Entonces, existe un cierto diagrama $\mathcal{D}_{\mathcal{G}} : G_{\mathcal{G}} \in S_{\mathcal{G}}$ en una superficie compacta $S_{\mathcal{G}}$ que tiene a $\mathcal{G}$, con la información descrita, como uno de sus posibles códigos.

Demostración: Fácil. Tras construir un entorno regular N_L de la lista formada por la yuxtaposición de las listas del código y construir una cara regular con los invariantes descritos por cada nodo de cara, se pegan los unos a los otros usando el resto de la información. □

Para averiguar cuál es la superficie $S_{\mathcal{G}}$ en la que se realiza el diagrama $\mathcal{D}_{\mathcal{G}}$ definido por un cierto código, lo primero a tener en cuenta es que cada componente conexa del grafo $\mathcal{G}$ representa a una componente conexa de la superficie $S_{\mathcal{G}}$. Por tanto, podemos restringirnos al caso en que $\mathcal{G}$ y $S_{\mathcal{G}}$ sean conexos. En este caso, $S_{\mathcal{G}}$ es orientable si las listas de todos los nodos componente son orientables y una orientación elegida en una de ellas se puede transmitir de forma compatible a todas las caras regulares y a las demás componentes conexas del diagrama. La característica de Euler de $S_{\mathcal{G}}$, por su parte, se obtiene directamente por la fórmula $F - A + V$, salvo que F no es ahora el número de caras sino la suma de las características de Euler de éstas, que se obtienen de sus invariantes.

De nuevo, la realizabilidad de todo código como diagrama y la unicidad de ésta realización son de utilidad a la hora de construir un catálogo de tipos combinatorios de diagramas. La forma de decidir si dos códigos diferentes producen diagrams equivalentes o no, se trata en la próxima Sección, aunque el método es más complicado que en el caso celular.

1.3.2 Código canónico. El caso de que el grafo de componentes sea un árbol

El problema de decidir si dos códigos dados representan diagramas equivalentes o no es más complejo que el problema equivalente para listas/equivalencia de diagramas celulares.

Para empezar, obsérvese que la afirmación de que dos diagramas sean equivalentes implica que sus grafos de componentes conexas sean isomorfos. Por otro lado, cualquier grafo bipartito de tamaño l puede aparecer como el grafo de componentes conexas de un diagrama de tamaño l. Ésto es consecuencia del Teorema 1.3.3 si tomamos como lista de cada nodo de componente una cuyo diagrama regular tenga

tantas caras como aristas. Por ejemplo, la del diagrama formado por una unión de meridianos de una esfera.

Puesto que no se conocen algoritmos polinomiales para decidir si dos grafos (ni siquiera bipartitos) son isomorfos, no podemos esperar poder decidir en tiempo polinomial si dos diagramas son equivalentes. (Para información sobre la complejidad de decidir isomorfismo de grafos existe la monografía [Koebler-S-T] o se puede consultar [Garey-Johnson]. El problema, más general, de decidir si un cierto grafo es isomorfo a algún subgrafo de otro es NP-completo. El problema de isomorfía de grafos en sí no está probado ni refutado que lo sea).

Sin embargo, aún podemos tener alguna esperanza si fijamos la superficie S en la que están inmersos nuestros diagramas, gracias a los dos lemas que siguen:

Lema 1.3.5 *Sea $\mathcal{D} : G \to S$ un diagrama en una cierta superficie conexa S. Sea $\mathcal{G}$ el grafo de componentes conexas de $\mathcal{D}$. Sea a la arista de $\mathcal{G}$ que representa a un cierto ciclo de borde B_a del diagrama. Entonces, la eliminación de la arista a en $\mathcal{G}$ desconecta al grafo si y sólo si recortar el ciclo de borde B_a desconecta a la superficie S.*

Demostración: Sea f la cara unida al ciclo de borde B_a. Consideremos el resultado de recortar la superficie S por el ciclo de borde B_a y pegar dos círculos, uno al entorno regular y otro al complemento regular. Su único efecto en el código es el de desdoblar la cara f en dos, una con la misma orientabilidad y género que f (pero con una componnte menos en su borde) y otra celular. Además, la primera mantiene todas las conexiones con ciclos de borde que tenía f, salvo por el ciclo B_a y la segunda está sólo conectada a B_a. Traduciendo al grafo de componentes conexas, el resultado es el de desconectar la arista a de la componente de cara de f y dejerla conectada a un nodo de cara nuevo, que no tiene más conexiones. Ésto demuestra el Lema. □

Lema 1.3.6 *Sea $\mathcal{D} : G \to S$ un diagrama en una cierta superficie conexa S. Sea $\mathcal{G}$ el grafo de componentes conexas de $\mathcal{D}$. Entonces, el número de ciclos disjuntos del grafo $\mathcal{G}$ (esto es, el número máximo de aristas que se pueden cortar sin desconectarlo) es, a lo más, la parte entera de $1 - \chi/2$, donde χ es la característica de Euler de S.*

Demostración: Supongamos que el grafo $\mathcal{G}$ posee un número k de aristas que, si se eliminan, no lo desconectan. Entonces S posee un número k de componentes de borde que, si se recortan, no la desconectan.

Ahora bien, los ciclos de borde de un diagrama son inmersiones de una circunferencia con un entorno orientable, ésto es, *óvalos* según la terminología que introduciremos en la Sección 2.1 para el plano proyectivo. La operación de cortar la superficie por uno de ellos y añadir dos círculos hace aumentar en dos unidades la característica de Euler. Si hacemos ésto a los k que tenemos, obtendremos una cierta superficie que es aún conexa y, por tanto, $\chi + 2k \leq 2$ □

Para grafos con un número máximo t de aristas que no los desconectan sí existen algoritmos polinomiales en su tamaño l para decidir isomorfismo. Por ejemplo, dados dos tales grafos hay del orden de l^t formas diferentes de convertirlos en árboles

cortando, a lo más, t aristas. Para cada una de las formas la equivalencia de los árboles se puede decidir en tiempo $O(l \log(l))$. Ésto poduce un algoritmo de complejidad, esencialmente, l^t.

Éste mismo algoritmo serviría, con pequeños añadidos, para decidir si dos diagramas generales son equivalentes, supuesto que conozcamos como decidir la equivalencia de diagramas para el caso en el que el grafo de componentes conexas sea un árbol. Es en este caso en el que nos vamos a concentrar a partir de ahora, no sólo por esta razón, sino también porque engloba a las tres superficies esenciales en la que ocurren las aplicaciones que nos interesan. A saber, la esfera, el plano real y el plano proyectivo.

Supongamos entonces que el grafo de componentes conexas $\mathcal{G}$ de un cierto diagrama $\mathcal{D}$ es un árbol. Nuestro objetivo es encontrar una forma canónica para el código de $\mathcal{D}$. Si elegimos uno de los nodos de $\mathcal{G}$ como raíz del árbol, la estructura de árbol con raíz del mismo se puede representar como una lista de listas, donde el anidamiento de listas representa la jerarquía en el árbol. La información que, junto con el árbol de componentes, constituye el código, deberá también ir incluida como parte de esta lista, en el lugar en el que corresponda. Ésto significa que la lista tendrá una estructura que permita decidir qué trozo de información está en cada lugar, pero no merece la pena que detallemos la manera en que ésto se hace. De hecho, nuestro algoritmo será independiente de este detalle (salvo que las comparaciones lexicográficas, por supuesto, dependen de cómo se organiza la información en la lista).

Hay un número finito de listas que pueden representar al árbol (para las distintas elecciones de la raíz y las distintas elecciones del orden en que aparecen los "hijos" de cada nodo) y un número finito de maneras de elegir la información del código (es decir, las listas de aristas, la orientación de las caras orientables y la semiarista que representa a cada ciclo de borde). Por tanto, podemos definir como código canónico aquél que produce la lista lexicográficamente más pequeña, como hicimos en la Sección 1.2.2 para las listas de aristas. Vamos a estudiar un algoritmo para encontrar el código canónico.

En primer lugar, vamos a deshacernos de los posibles puntos aislados del diagrama codificándolos no como componentes conexas adicionales del diagrama sino como un "invariante" más de la cara en la que están: cada cara tendrá asociado un cuarto invariante, que representa el número de puntos aislados en ella. En estas condiciones el número total de aristas l del diagrama es un buen parámetro para definir la "complejidad" del mismo, puesto que los números de caras, componentes conexas, ciclos de borde, vértices, etc. están acotados por una función lineal de l.

Proposición 1.3.7 *Sea $\mathcal{C}$ una lista que representa al código asociado a un diagrama $\mathcal{D}$ con l aristas cuyo grafo de componentes conexas es un árbol (con raíz, dada por $\mathcal{C}$). Supongamos que la raíz del código canónico asociado al diagrama coincide con la de $\mathcal{C}$. Entonces, existe un algoritmo que, con* input *$\mathcal{C}$, calcula el código canónico $\mathcal{C}_C$ asociada a $\mathcal{D}$ en tiempo $O(l^2)$.*

Demostración: El algoritmo será recursivo. A partir de $\mathcal{C}$ se obtendrán las sublistas que representan los subárboles que tienen como raíz a cada uno de los hijos de la raíz. De cada uno de ellos se obtendrá una versión canónica, usando la recursión.

Después, se compararán unos con otros y se reordenarán como hijos de la raíz del árbol en orden lexicográfico. La forma detallada de hacer ésto, y la demostración de la cota (que se hace de forma inductiva) es:

Si la raíz es un nodo cara de valencia k, consideraremos los diagramas $\mathcal{D}_1, \ldots, \mathcal{D}_k$ que resultan de sustituir dicha cara por k círculos en los k ciclos de borde. Sean $l_1, \ldots, l_k$ los números de aristas de cada uno de ellos ($\sum l_i = l$). Por hipótesis inductiva, podemos obtener los códigos canónicos de ellos en tiempo $\mathrm{O}(l_1^2 + \cdots + l_k^2) \leq \mathrm{O}(l^2)$. Una vez hecho ésto ordenemos los subcódigos (también en tiempo $\mathrm{O}(l^2)$, por el mismo método que en la parte final de la demostración de la Proposición 1.2.4) y recoloquémolos. Si la cara del nodo raíz es orientable, deberemos hacer ésto para cada una de las dos posibles orientaciones de la cara, obteniendo dos posibles códigos canónicos (que difieren sólo en la buene o mala orientación de sus ciclos de borde) y comparar los dos códigos obtenidos.

Si la raíz es un nodo componente de valencia k, consideraremos los diagramas $\mathcal{D}_1, \ldots, \mathcal{D}_k$ que resultan de sustituir dicha componente por k círculos en los k ciclos de borde. Sean $l_1, \ldots, l_k$ los números de aristas de cada uno de ellos. Ahora se tiene $\sum_{i=1}^{k} l_i = l - l_0$, donde l_0 es el número de aristas en la raíz. Obtengamos los códigos canónicos de cada uno de los $\mathcal{D}_i$ en tiempo total $\mathrm{O}(l^2)$ (de nuevo por hipótesis inductiva). Reordenemos lexicográficamente dichos códigos.

Para cada una de las $4l_0$ posibles elecciones de una semiarista inicial de la componente raíz y una orientación de su vértice, obtengamos la lista canónica correspondiente (en tiempo $\mathrm{O}(l_0)$ cada una). Para cada una de ellas rehagamos el código, decidiendo si los ciclos de borde de la componente raíz están bien orientados (de acuerdo con la lista en cuestión) y colocando los códigos de los diagramas $\mathcal{D}_1, \ldots, \mathcal{D}_k$ lexicográficamente ordenados como hijos de la componente raíz. Ésto se hace en tiempo $\mathrm{O}(l)$ para cada posible elección inicial, lo cual produce $\mathrm{O}(ll_0) \leq \mathrm{O}(l^2)$. □

Una pequeña observación, que parece contradecir a la Proposición 1.3.7. Sea S la superficie orientable de género $k_1 + k_2$ (un toro con $k_1 + k_2$ agujeros). Consideramos el diagrama formado por un óvalo que separa k_1 de los agujeros de los otros k_2. La obtención del código canónico implica decidir cuál de los dos números k_1 y k_2 es más pequeño. Por tanto, ¡podemos comparar números enteros arbitrariamente grandes en tiempo constante! La razón de ello es que nuestro "modelo de complejidad" supone que las operaciones aritméticas se hacen en tiempo constante (cf. las primeras observaciones de la demostración de 1.2.4). Si considerásemos un modelo más realista en el que se cuente la complejidad 'bit' de operaciones aritméticas, el parámetro l deberá tener en cuenta la complejidad topológica de las distintas caras y el número de puntos aislados del diagrama y, además, nuestra cota de complejidad se convertiría en $\mathrm{O}(l^2 \log(l)^t)$ para un cierto t que no entramos a analizar.

Estudiemos la aplicación de la Proposición 1.3.7 a los tres casos arriba mencionados, que son el plano afín, la esfera y el plano proyectivo. En el caso peor, para obtener el código canónico tendremos que aplicar el algoritmo descrito tantas veces como posibles elecciones hay de la raíz, lo cual produce un algoritmo de tiempo $\mathrm{O}(l^3)$. Sin embargo, en el caso del plano proyectivo podemos *postular* que la raíz del árbol del código canónico sea la única componente con entorno regular no orientable o la única cara no orientable. Todo diagrama en el plano proyectivo posee una, y sólo una, de las dos cosas. En el caso del plano afín, que consideramos

como una esfera con un punto distinguido, postularemos que la raíz del árbol sea la componente o la cara que contenga al punto del infinito. Nótese que todas las componentes no acotadas del diagrama en el plano afín se consideran como una sóla en el diagrama en la esfera, puesto que todas son incidentes sobre el vértice del infinito. Por tanto:

Corolario 1.3.8 *Sea $\mathcal{D}$ un diagrama con l aristas en el plano afín o proyectivo. Entonces, existe un algoritmo que, con* input *un código asociado a $\mathcal{D}$, calcula el código canónico asociado a $\mathcal{D}$ en tiempo* $O(l^2)$.

Corolario 1.3.9 *Sean $\mathcal{D}_1$ y $\mathcal{D}_2$ dos diagramas con l aristas en el plano afín o proyectivo. Entonces, existe un algoritmo que, con* input *sendos códigos asociados a $\mathcal{D}_1$ y $\mathcal{D}_2$, decide si $\mathcal{D}_1$ y $\mathcal{D}_2$ son equivalentes en tiempo* $O(l^2)$.

1.4 Aplicación al caso de curvas algebraicas planas reales

1.4.1 Antecedentes

El estudio que hemos llevado a cabo en este Capítulo tiene una aplicación natural a la codificación de la topología de curvas algebraicas planas reales (ya sean afines o proyectivas). Como haremos también en el Capítulo 2, usaremos el término curva algebraica como abreviatura de curva algebraica plana real (nótese, de todas formas, que allí nos referiremos siempre a curvas proyectivas).

Es bien sabido que una curva algebraica real es una variedad diferenciable (en particular, topológica) en todos sus puntos salvo un número finito de ellos y que en éstos es localmente homeomorfa al entorno de un vértice de valencia par de un grafo geométrico. Ésto, unido al hecho de ser cerradas y a una condición de finitud de ramas al infinito (para las curvas afines), implica necesariamente el que sean diagramas en el sentido de nuestra definición. Debemos mencionar que es esta aplicación al caso de curvas algebraicas la que ha originado y dirigido nuestro trabajo en este problema.

El problema del cálculo de la topología de una curva algebraica plana real ha sido tratado ampliamente de forma algorítmica en los últimos 15 años. Citaremos sólo algunos de las referencias más significativas, como son [Arnon-C-McC], [Cellini-G-T] o [Roy]. En todos los casos la estructura del algoritmo es similar, aunque los aspectos técnicos varían de uno a otro. La idea central consiste en dividir el plano en franjas verticales en cada una de las cuales la topología de la curva es trivial: la de un cierto número de líneas que la recorren de izquierda a derecha. Los rectas de separación entre una franja y otra son de la forma $\{X = a_i\}$ para las distintas raíces a_i del discriminante del polinomio que define la curva y su derivada respecto a la dirección vertical. La topología de la curva viene caracterizada por qué ramas de las que atraviesan cada franja van a parar a qué puntos de las rectas entre franjas.

Además de diseñar algoritmos para producir esta descomposición del plano (a la que llamaremos Descomposición Cilíndrica, aunque éste es un concepto más general) los mencionados autores se han preocupado de diseñar formas adecuadas de codificar la topología de la curva con el fín último de, al menos, ser capaces de decidir si dos curvas dadas son topológicamente equivalentes o no (en el sentido de nuestra

definición o alguno muy parcido). Sin embargo, los métodos exhibidos sólo eran totalmente satisfactorios para el caso de curvas no-singulares en el plano proyectivo. En este caso, la codificación puede hacerse siplemente mediante un árbol, similar a nuestro árbol de componentes conexas, al que no es necesario añadir información extra. El caso de curvas singulares es mencionado por [Cellini-G-T] como "más complicado y todavía no del todo claro".

El único intento que conocemos de dar una solución completa al caso singular es el de [G.Corbalán-Recio] (cf. también [G.Corbalán]) los cuales daban de hecho un algoritmo de decisión de si dos curvas algebraicas en el plano afín son equivalentes o no. Sin embargo, el algoritmo era de complejidad exponencial, sólo se aplicaba al caso de puntos dobles y era demasiado complicado como para intentar una implementación. Mencionemos, de todas formas, que dichos métodos constituyeron el punto de partida para nuestro trabajo.

Con los métodos descritos en las Secciones anteriores (en particular, los algoritmos de obtención de la lista y el código canónico) es relativamente sencillo decidir esta equivalencia topológica de curvas algebraicas singulares. La presente Sección está destinada a presentar una implementación de un algoritmo que hace esta decisión, para curvas algebraicas afines. En un futuro próximo la implementación se extenderá al caso de curvas proyectivas.

Xtopo [Hong] es un programa especialmente eficiente desarrollado por Hoon Hong del Instituto RISC-Linz que, con INPUT un polinomio f en dos variables y de coeficientes enteros, calcula la topología de la curva algebraica $f(x, y) = 0$, tanto en el plano afín como el proyectivo por medio de una descomposición cilíndrica. Nuestro trabajo ha sido tomar el OUTPUT producido por Xtopo (que es una descripción natural de la descomposición cilíndrica), traducirlo a un código del diagrama subyacente en el sentido de nuestra Sección 1.3 y obtener de él el código canónico detallado en 1.3.2. El programa está escrito en C, utilizando la librería SACLIB [B-C-E-H-J-K-L-N] de álgebra computacional y el sistema de ventanas X Windows. En la implementación ha colaborado Mark J. Encarnación, del mismo Instituto RISC-Linz y ésta ha sido presentada en [Enc.-Hong-Santos].

Además de producir el código canónico y de decidir equivalencia topológica de las curvas definidas por los polinomios que se le dan como INPUT, el programa produce el dibujo de un grafo topológicamente equivalente a la curva (con las componentes conexas afines separadas en diferente color), junto a otro dibujo de la curva obtenido numéricamente en el que se aprecia su geometría, aunque pueden perderse detalles topológicamente significativos.

1.4.2 Nuestra implementación

El OUTPUT natural producido por Xtopo, como el de cualquier algoritmo que utilice una heurística de descomposición cilíndrica, consistía en una lista con tantos elementos como líneas verticales hay entre las franjas de la descomposición. Cada elemento es a su vez una lista que indica, para cada punto en la vertical y en orden ascendente, cuántas ramas inciden sobre dicho punto a su derecha y a su izquierda.

Dicho OUTPUT ha sido traducido a un código, similar a los de nuestra Sección 1.3, un tanto más simple por tratarse de diagramas en la esfera. Por ejemplo, los nodos cara no aparecen en el árbol, porque la información que contienen es trivial: cada cara regular es homeomorfa a una esfera con tantos agujeros como componentes

del diagrama son adyacentes a ella. Además, la información sobre la semiarista que señala a un ciclo de borde es diferente: en la implementación, el código canónico contiene todas las semiaristas que tienen el ciclo de borde a la izquierda, ordenadas cíclicamente. Por último, en vez de calcular todos los ciclos de borde, se calculan sólo el de la cara exterior de cada componente y aquéllos que corresponden a caras que contienen alguna componente interior. Los primeros se denominan OUTER FACE en el código y los segundos CONTAINING FACE, y ambos se consideran parte del código de la componente interior.

La obtención del árbol de componentes conexas de la Sección 1.3.2 a partir de dicho OUTPUT requiere los siguientes pasos (véase la página 39).

(i) S = CODE(HHCODE(T)): Construye una lista global de los puntos P y aristas E de la descomposición.

Para ello, se denota a cada punto de la curva en una de las líneas verticales de la descomposición con dos índices (i, j), que quieren decir que dicho punto es el j-ésimo punto empezando por abajo de la i-ésima línea empezando por la izquierda. Como estamos trabajendo en el plano afín debe añadirse un punto, que denotamos $(0, 1)$, el cual representa al infinito. Constrúyase la lista P con esos puntos. Después, construyamos una lista E de las aristas de la descomposición, donde cada arista es un par de puntos $P_1 = (i_1, j_1)$ y $P_2 = (i_2, j_2)$ unidos de acuerdo con la información que se nos ha dado. Las semiaristas alrededor de un vértice son numeradas en sentido horario comenzando debajo del punto (dar la misma orientación a todos los puntos nos evitará tener que tratar con aristas mal orientadas). A cada punto P de la lista se le añade un tercer número que indica su valencia en el grafo.

Algunos de los puntos (i, j) que aparecen en esta etapa no son topológicamente significativos, es decir, tienen valencia dos en el grafo. En la última etapa los eliminaremos, para codificar realmente la topología, pero de momento nos conviene mantenerlos porque la disposición especial de los vértices y aristas de la descompósición cilíndrica será de ayuda en algunos pasos intermedios.

(ii) C = SEP_PROJ(S): Separa las listas P y E en componentes conexas, incluyendo la información de la cara exterior.

Las listas P y E se separan en componentes conexas $P_1, \ldots, P_K$ y $E_1, \ldots, E_K$, usando un algoritmo típico para separar componentes conexas de un grafo (V, E). Estas componentes conexas son las componentes del diagrama "en la esfera", lo que significa que todas las componentes no acotadas se consideran conectadas entre sí por el punto del infinito.

(iii) F = FACES(C): Obtención de la cara exterior de cada componente.

Para cada componente conexa (P_k, E_k) (excepto la que contiene al punto del infinito etiquetado $(0, 1)$), construimos la lista f_k de todas las semiaristas que tienen a la cara exterior a su izquierda (usando la función FACE). Para ello, tomamos como semiarista inicial la más pequeña (lexicográficamente) de la lista y reconstruimos la cara a su izquierda por un proceso como el descrito al final de la Sección 1.2.4. Aquí estamos aprovechando el hecho de que los puntos están indexados de manera ordenada en columnas.

A la componente no acotada se le asigna como información la lista de las ramas al infinito. Ésto no tiene sentido como ciclo de borde, pero será usado a la hora de calcular el código canónico.

(iv) B = TREE(F) Construcción del árbol de componentes conexas.

La raíz del árbol es la componente (P_0, E_0) que contiene al punto del infinito $(0,1)$. Si ninguna componente tiene a dicho punto (es decir, si la curva era acotada), tomamos $P_0 = (0,1)$ y E_0 vacío. Las componentes sucesivas se añaden al árbol, usando la siguiente estrategia:

Considérese en cada momento la componente $A_k = (P_k, E_k)$ que contiene al punto (i,j) lexicográficamente más pequeño de entre las componentes que aún no han sido añadidas al árbol. Esta componente estará sólo contenida en componentes que ya hayan sido añadidas al árbol. Para buscar cuál de ellas es su "padre" (función FATHER) comenzamos por la raíz y vamos descendiendo en el árbol, comprobando que la componente a la que descendemos contenga a la que queremos colocar (mediante la función CONTAINS). Cuando no sea posible descender más, habremos terminado.

TREE termina calculando la cara del padre de la componente en la que la componente está contenida (función CFACE, similar a FACE).

(v) B = SIMPLIFY(B): Finalmente, elimina los puntos no singulares del código.

Para cada punto P del código con valencia dos, busca las dos aristas que lo contienen y las reemplaza por la que resulta de unirlas (REM_REG). De igual modo, elimina las semiaristas correspondientes de las listas OUTER FACE y CONTAINING FACE de cada componente (REM_REG_OR). Aquí aparece otra diferencia de la implementeción con lo descrito a lo largo del capítulo: cuando una componente no contenga ningún vértice de valencia diferente de dos, su lista final será vacía. Ésto no produce ambigüedad porque tal componente es, necesariamente, un óvalo.

El paso más complicado del algoritmo es el (iv) y es el único del que analizaremos la complejidad. La complejidad que damos a continuación no es la de nuestra implementación, porque requeriría incluir punteros entre aristas que permitan encontrar la arista que contiene a una cierta arista en tiempo constante. En cualquier caso, la complejidad asintótica del caso peor en nuestra implementación seguirá siendo polinomial de grado bajo.

El caso peor implicará decidir si cada una de las componentes A_k está contenida en alguna otra A_l (función CONTAINS). Cada decisión se hace contando el número de cortes de una semirrecta vertical que parte de un punto de A_k con el ciclo de aristas exteriores de A_l, en tiempo lineal en la longitud de este ciclo. Por tanto, el tiempo necesario para añadir la componente A_k al árbol es $O(n_E)$, donde n_E representa el número total de aristas de la descomposición, y el tiempo para construir el árbol completo es $O(n_E K)$, siendo K el número de componentes conexas. La función CONTAINS utiliza fuertemente el hecho de que las aristas forman una descomposición cilíndrica.

Obsérvese que, en el código obtenido, cada punto está aún etiquetado por sus dos índices (i,j), que provienen de la descomposición cilíndrica. Puesto que el

código aún no es canónico, los "nombres" dados a cada vértice no tienen ninguna relevancia.

La obtención de un código canónico a partir del obtenido con este algoritmo se efectua en la función CANONIZE. Esta función se aplica de manera recursiva a las componentes-hijo de la raíz. Para cada componente y cada elección de la semiarista inicial se hace el proceso MAKE_DICT, que construye la nueva lista de aristas y un diccionario que traduce la antigua a la nueva, como se describió en la Sección 1.2.2. El diccionario se mantendrá como parte del código, porque sirve de ayuda para reconocer cómo se corresponde el código canónico con el dibujo producido por Xtopo. Sin embargo, el diccionario contiene información "no canónica", la cual no debe ser considerada para las comparaciones lexicográficas. Por esta razón, todas las comparaciones se hacen con una función especial TREECOMP que extrae la información relevante al código canónico y deshecha la demás. Las funciones TRANSLATE_ITEM y TRANSLATE_FACE traducen, respectivamente, una semiarista y una lista de semiaristas a través del diccionario.

Finalmente, la función TWRITE escribe la información del árbol de componentes conexas en pantalla, con su estructura de árbol reflejada por la sucesiva indentación.

De nuevo, la implementación realizada no hace el uso de punteros descrito en la demostración de la Proposición 1.2.4, gracias al cual se obtenía la cota de complejidad $O(l^2)$ para el algoritmo. Los algoritmos implementados tienen por tanto una complejidad superior. La razón por la que no hemos intentado optimizar la implementación es que nuestro proceso necesita ser llevado a cabo una sola vez para cada curva algebraica y su complejidad, aún sin optimizar, es bastante inferior a la de efectuar la descomposición cilíndrica como paso previo. En los ejemplos con los que hemos experimentado, el cálculo del código y su forma canónica ocupa siempre menos del 5% del tiempo total.

Por otra parte, se han hecho algunas mejoras específicas para al caso del plano afín, como son el hecho de que en las listas canónicas se toma siempre como semiarista inicial una de las más exteriores (o una de las que van al infinito, para la componente no acotada) y se ha fijado una orientación global del plano a priori (lo cual permite probar una sóla vez, y no dos, con cada semiarista). En el caso peor el número de semiaristas exteriores coincide con el total, pero en la mayoría es mucho menor.

Como subproducto del algoritmo se obtiene una información interesante sobre la simetría de cada componente del diagrama, la cual se incluye en el OUTPUT bajo el epígrafe CYCLIC SYMMETRY. Ésta información no es otra cosa que el número posible de elecciones de la semiarista inicial que producen el mismo código. Éste número coincide con el número de automorfismos del pseudo-grafo natural de la componente conexa que se pueden extender a un homeomorfismo global del plano que conserva la orientación.

1.4.3 Algunos ejemplos

Terminamos esta Sección con dos sesiones de ejemplos con Xtopo. En la primera el INPUT es un fichero que contiene 30 polinomios de dos variables y de grados 4 y 5. Los dos primeros producen las descomposiciones cilíndricas de la Figura 1.8 y el siguiente texto en la pantalla:

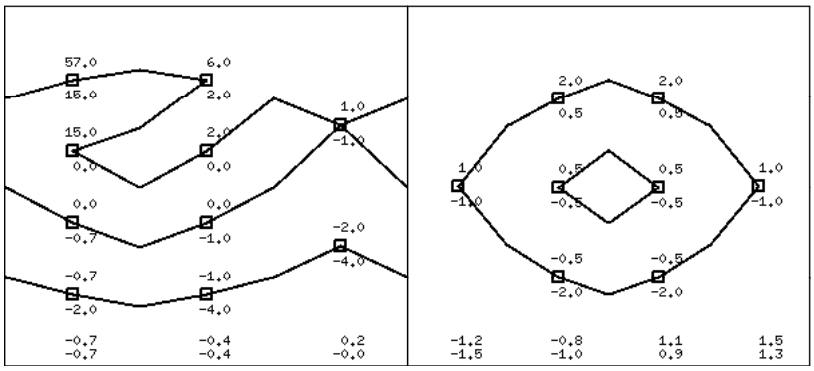


Figura 1.8: Descomposiciones cilíndricas de las dos primeras curvas del ejemplo.

```
Reading in polynomial number 1

y^5 + 37 x^5 y^4 + 95 x^3 y^4 - 21 x^3 y^3 - 11 x y^2 + 52 y^2
+ 45 x^5 y - 18 x^2

Analyzing the topology...
   684 milli-seconds for computing the topology



            THIS IS THE CANONICAL CODE:
Cyclic symmetry: 1
Points: ((1,((0,1),6,6)),(2,((3,2),4,4)))
Edges: (((1,1,6),(1,2,6)),((1,3,6),(2,1,4)),((1,4,6),(2,4,4)),
((1,5,6),(2,3,4)),((1,6,6),(2,2,4)))
Outer Face: ((1,1,6),(1,2,6),(1,3,6),(1,4,6),(1,5,6),(1,6,6))
NO SONS

   0 milli-seconds for computing the canonical code



==========================================
Reading in polynomial number 2



y^4 + 2 x^2 y^2 - 3 y^2 + x^4 - 3 x^2 + 2



Analyzing the topology...
   67 milli-seconds for computing the topology
```

```
             THIS IS THE CANONICAL CODE:
Cyclic symmetry: 1
Points: ((1,((0,1),0,0)))
Edges: ()
Outer Face: ()
Sons:
   Containing face: ()
      Cyclic symmetry: 1
      Points: ()
      Edges: (())
      Outer Face: ()
      Sons:
         Containing face: ()
            Cyclic symmetry: 1
            Points: ()
            Edges: (())
            Outer Face: ()
            NO SONS

   0 milli-seconds for computing the canonical code
   0 milli-seconds for comparing
==========================================
```

El código canónico debe ser leído de la siguiente manera: para cada componente conexa (recuérdese que todas las componentes no acotadas se consideran una sola y que siempre hay una componente no acotada, quizás vacía) produce cinco informaciones: la simetría cíclica, que ya ha sido discutida, una lista de puntos, una de aristas, una de semiaristas exteriores y un mensaje que nos dice si la componente tiene alguna otra en una de sus caras interiores (ésto es, algún hijo en el árbol). Si los tiene, éstos aparecen indentados, en el mismo formato, y precedidos de la lista de semiaristas de la cara que los contiene.

Las semiaristas que forman parte de una arista o de una de las listas de semiaristas tienen asignados tres números. Los dos primeros son el número que corresponde al vértice y a la semiarista. El tercero es la valencia del vértice en el grafo.

La lista de puntos es, de hecho, el diccionario que se utiliza para traducir cada semiarista del código canónico a una de las de la descomposición cilíndrica. Más precisamente, cada elemento del diccionario tiene la forma $(P,((i,j),b,m))$, indicando que el punto P del código canónico corresponde con el punto (i,j) de la descomposición cilindrica y que la semiarista etiquetada "1" en el código canónico es la b-ésima que se encuentra alrededor del vértice, comenzando debajo de él y girando en el sentido de las agujas del reloj. El último número, m, de nuevo representa la valencia del vértice. La razón por la que esta valencia aparece repetida tantas veces es que es necesaria para todas las operaciones aritméticas con los índices de las semiaristas, que son módulo m.

Obsérvese que la raíz del árbol en la segunda curva (que representa a una componente no acotada "vacía") tiene un punto pero ninguna arista. Ésto significa que la componente no acotada está siendo considerada como un punto aislado en el infinito. En cambio, los dos óvalos anidados se representan por componentes del

árbol que tienen vacías tanto sus listas de puntos como de aristas.

Continuando con los ejemplos, cuando se llega a un polinomio que define una curva topológicamente equivalente a alguna de las anteriores, el programa nos avisa de este hecho. Por ejemplo, el polinomio número 22 produce el dibujo de la Figura 1.9 y el siguiente texto:

```
Reading in polynomial number 22

y^5 - 17 x^5 y^4 - 30 x^5 y^3 - 97 x y^2 + 124 x^5 y - 101 x^4 y
- 73 x^4 - 99 x^2

Analyzing the topology...
   517 milli-seconds for computing the topology

            THIS IS THE CANONICAL CODE:
Cyclic symmetry: 1
Points: ((1,((0,1),1,6)),(2,((1,1),4,4)))
Edges: (((1,1,6),(1,2,6)),((1,3,6),(2,1,4)),((1,4,6),(2,4,4)),
((1,5,6),(2,3,4)),((1,6,6),(2,2,4)))
Outer Face: ((1,1,6),(1,2,6),(1,3,6),(1,4,6),(1,5,6),(1,6,6))
NO SONS

   0 milli-seconds for computing the canonical code
THIS CURVE HAS THE SAME TOPOLOGY AS CURVE NUMBER 1
THIS CURVE HAS THE SAME TOPOLOGY AS CURVE NUMBER 4
   0 milli-seconds for comparing
```

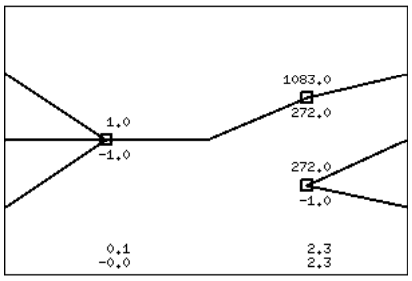


Figura 1.9: Descomposición cilíndrica de una tercera curva, topológicamente equivalente a la segunda de la Figura 1.8.

En estos ejemplos pequeños el tiempo de ejecución de la codificación es demasiado pequeño para ser expresado en milisegundos. En una segunda sesión, vamos a introducir una curva de grado 12 con topología complicada y otra que se obtiene de ella girando 90 grados, lo cual equivale a intercambiar las variables x e y por ser la curva simétrica respecto a una recta vertical (ver la Figura 1.10). Las dos curvas son obviamente equivalentes, pero sus descomposiciones cilíndricas son diferentes. La segunda tiene del orden del doble de celdas que la primera. El texto producido por el programa es (se omite el código canónico, extremadamente largo):

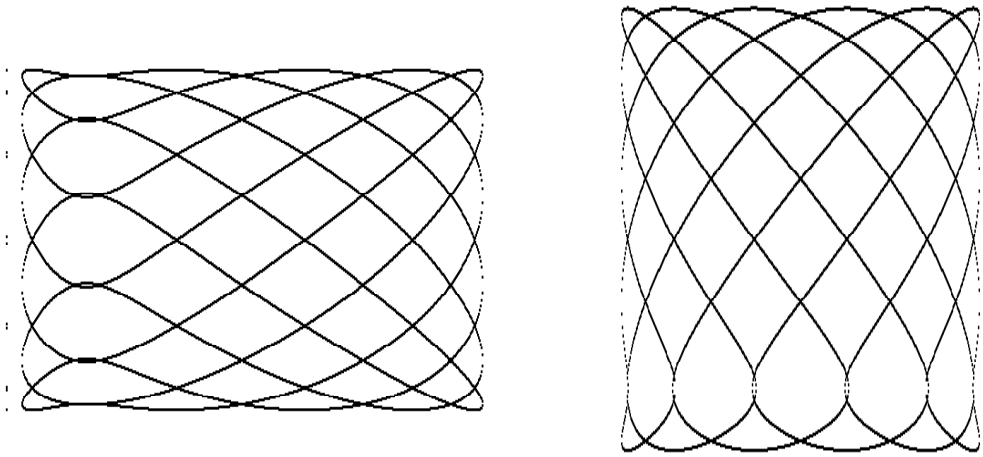


Figura 1.10: Una curva de grado 12 y la que resulta intercambiando las variables.

```
==========================================
A degree 12 curve with very complicated topology:

Reading in polynomial number 1

27000 y^12 - 648000 y^10 + 5832000 y^8 - 24192000 y^6 + 45360000 y^4
- 31104000 y^2 + 8 x^12 - 12 x^11 - 690 x^10 + 695 x^9 + 22890 x^8
- 12972 x^7 - 361432 x^6 + 83520 x^5 + 2681280 x^4 - 129600 x^3
- 7516800 x^2 + 6912000

Analyzing the topology...
   66717 milli-seconds for computing the topology

   1483 milli-seconds for computing the canonical code


==========================================
The same curve, with variables changed:

Reading in polynomial number 2

8 y^12 - 12 y^11 - 690 y^10 + 695 y^9 + 22890 y^8 - 12972 y^7
- 361432 y^6 + 83520 y^5 + 2681280 y^4 - 129600 y^3 - 7516800 y^2
+ 27000 x^12 - 648000 x^10 + 5832000 x^8 - 24192000 x^6
+ 45360000 x^4 - 31104000 x^2 + 6912000

Analyzing the topology...
   39967 milli-seconds for computing the topology

   1800 milli-seconds for computing the canonical code
THIS CURVE HAS THE SAME TOPOLOGY AS CURVE NUMBER 1
```

```
   0 milli-seconds for comparing

==========================================
```

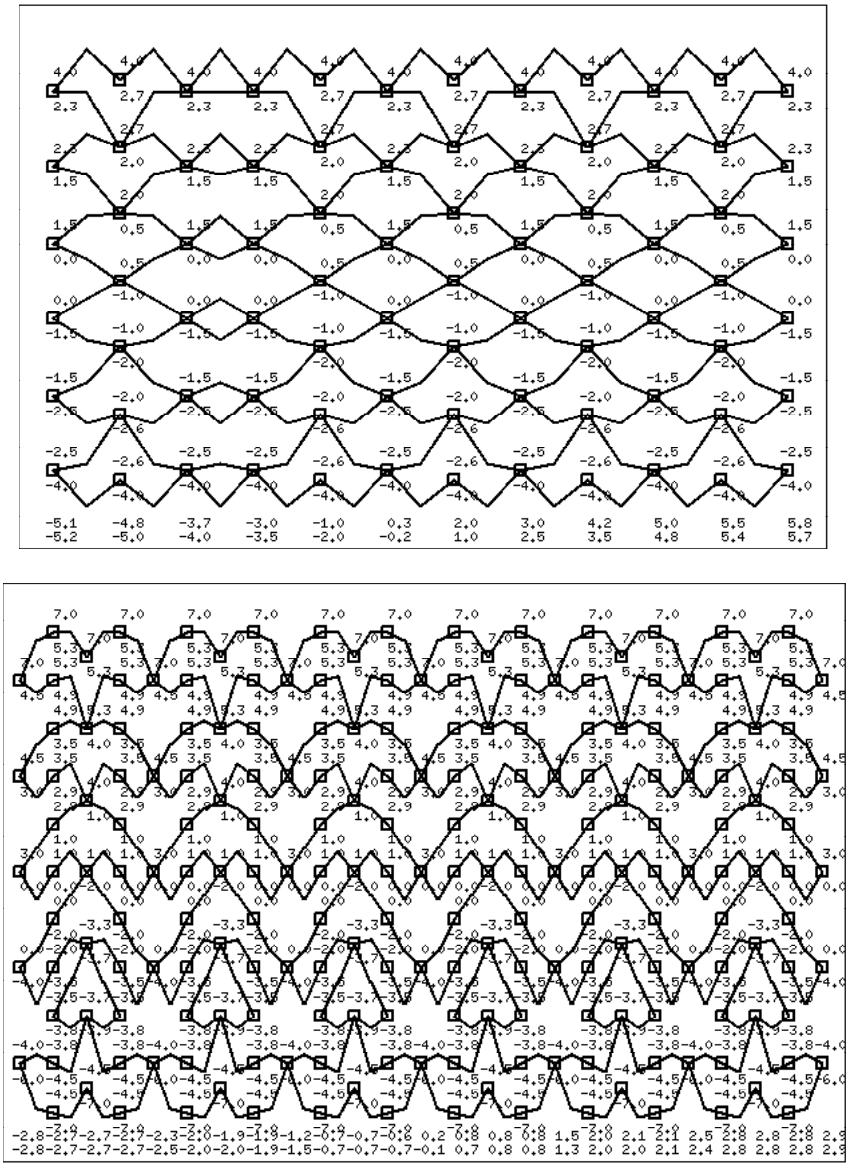


Figura 1.11: Descomposiciones cilíndricas de las curvas de la Figura 1.10.

Es decir, el programa reconoce que las dos curvas son topológicamente equivalentes. El tiempo necesario para calcular el código canónico es pequeño, comparado con el necesario para calcular la descomposición cilíndrica, y el tiempo que toma comparar dos códigos canónicos es insignificante.