UNIVERSIDAD
DE GRANADA

LIBRERÍA EN PYTHON DE TEORÍA DE GRUPOS

ALBERTO JESÚS DURÁN LÓPEZ

Trabajo Fin de Grado

Doble Grado en Ingeniería Informática y Matemáticas

Tutores

Manuel Bullejos Lorenzo

Pedro Abelardo García Sánchez

FACULTAD DE CIENCIAS

E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, a 25 de noviembre de 2020

RESUMEN

Palabras clave: grupo, grupo libre, presentación, producto semidirecto, Problema de Palabras, Todd Coxeter, Python.

La Teoría de Grupos es un área de las Matemáticas que estudia la estructura algebraica de conjuntos dotados de diferentes operaciones binarias que satisfacen unos axiomas específicos. Sus aplicaciones van más allá de las Matemáticas, desde lo más profundo de la Física o Mecánica hasta las estructuras moleculares de la Química, y potencialmente aplicable a todos los estados y situaciones en los que la simetría intervenga.

En este proyecto se hablará sobre el concepto de Grupo y los axiomas que deben cumplir sus elementos. Además, se introducirán diferentes grupos junto a las propiedades que caracterizan a cada uno de ellos, como el grupo de las Permutaciones, Diédrico o grupo de los Cuaternios. Asimismo, presentaremos los grupos mediante una serie de generadores y relatores, es decir, dando lugar al concepto de *presentación de grupo*. Para ello se profundizará sobre grupos libres, concluyendo con el Problema de Palabras y describiendo el *Algoritmo de Todd Coxeter*, que trata de resolver este problema mediante la enumeración de clases.

Por otro lado, y usando toda la teoría introducida anteriormente, se llevará a cabo la construcción del producto semidirecto de grupos: una alternativa al producto directo (cartesiano) de grupos en el que se hará uso de acciones de grupo. Usando esta noción de producto semidirecto, y con la ayuda de los *Teoremas de Sylow* 7, se describirán las principales características y técnicas para la clasificación de grupos de orden n para algunos n específicos, ofreciendo así una alternativa a la clasificación usual.

En tercer y último lugar, se realizará una optimización y ampliación de la librería de Pedro A. García y José L. Bueso, basada en la librería de Naftali Harris [Har12], que recoge los aspectos más importantes de la Teoría de Grupos. A esta librería se incorporará una implementación del *Algoritmo de Todd Coxeter*, donde dados dos grupos G y $H \leq G$, se obtendrá una tabla de clases laterales que refleje la acción de G sobre G/H. Como consecuencia, no sólo obtendremos el índice $[G : H]$, sino que podremos dar estructura de grupo de Permutaciones a grupos definidos por una presentación.

Se ha realizado un tutorial en Jupyter mostrando diferentes ejemplos de ejecución, y la librería está disponible en https://github.com/lmd-ugr/Grupos.

ABSTRACT

Key words: group, free group, presentation, semidirect product, word problem, Todd Coxeter, Python.

Group Theory is an area of Mathematics that studies the algebraic structure of sets endowed with different binary operations and that satisfy specific axioms. Its applications go beyond Mathematics, from the depths of Physics or Mechanics to the molecular structures of chemistry, and potentially applicable to all states and situations in which symmetry intervenes.

In this project, we will deal with the concept of Group and the axioms that its elements must fulfill. In addition, different groups will be introduced in a first introductory chapter together with the properties that characterize each one of them, such as the Permutation, Dihedral or the Quaternion group. We will also present the groups through a series of generators and relators, that is, resulting in the concept of *group presentation*. In order to do this, we must talk about free groups and carry out their construction based on a set X, developed by Dyck.

One of the first problems that arises in giving a group through a presentation is that of determining when two elements of the group (given as words in the generators) are equal; that is, determine using the relators if two words in the generators give rise to the same element. This problem is known as the Word Problem and it first arose in 1911, by Max Dehn. Along with this problem, Dehn published an article with other problems, the Conjugation Problem and the Isomorphism Problem, being today the three best known decision problems in Group Theory.

Nowadays there are algorithms that may be able to solve the Word Problem for a group G defined by a finite presentation. The best known is called *Todd Coxeter Algorithm* which tries to solve this problem by a technique called coset enumeration of G/H, where G is a finite group defined by a presentation and H is a subgroup of it. An implementation of this algorithm has been incorporated into the library and numerous methods have been implemented that will allow us to obtain the index $[G : H]$ and a representation by permutations of G, among others. The latter is a key concept since we can define any group using the library through a presentation. Once the group is defined, the different methods implemented can be called to try to establish an isomorphism with a known group. We will detail the algorithm in section 2.3.1.1, and its implementation in section 5.3.1.

On the other hand, the construction of the semidirect product of groups will be carried out using all the theory introduced above. This tool is an alternative to the direct product (cartesian) of groups in which group actions are used. Using this notion of semidirect product, and with the help of *Sylow Theorems* 7, the main characteristics and techniques for the classification of groups of order n for some specific n will be described, thus offering an alternative to the usual classification. Carrying out this classification is a costly and difficult process. As it will be seen in later sections, not every group can be expressed as a semidirect product. For this reason, a study similar to the one Hölder did will be carried out, studying groups that follow similar patterns, where we will focus on groups whose order is the product of prime numbers, specially in these of order p, p^2, $2p$, pq and p^3, where p is a prime.

Moreover, a study will be conducted about the optimization and ampliation of Pedro A. García and José L. Bueso library, based on the Naftali Harris [Har12], which collects the most important aspects of Group Theory. In this optimization, new methods will be added and the main groups will be implemented in classes in order to equip each class with the operations that define the different groups. One of the main problems of the library was the impossibility of defining a group given by a presentation, that is, a group could be defined as from a Set and its binary operation but not from a set of generators and relators; as a consequence an implementation of the *Todd Coxeter Algorithm* will be incorporated into this library as we have already commented previously.

Finally, the documentation will be completed and a tutorial will be provided in Jupyter showing with different examples the use of the different methods and files that make up the library. The library is available at https://github.com/lmd-ugr/Grupos.

Declaración de originalidad

D. Alberto Jesús Durán López.

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2020/2021, es original; entendida esta, en el sentido de que no se ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada, a 25 de noviembre de 2020.

Fdo: Alberto Jesús Durán López.

AGRADECIMIENTOS

A mi familia, por su apoyo incondicional durante todos estos años. A mis amigos, por su compañía. A mis tutores Manuel y Pedro, por su ayuda y dedicación.

ÍNDICE GENERAL

GLOSARIO DE TÉRMINOS

$\lvert G\rvert$	*Orden del grupo G*
$\lvert g\rvert$	*Orden del elemento g*
$\mathbb{Z}_n$	*Grupo cíclico de orden n, notación aditiva*
C_n	*Grupo cíclico de orden n, notación multiplicativa*
F	*Grupo libre*
S_n	*Grupo de Permutaciones de n elementos*
A_n	*Grupo Alternado de S_n*
D_n	*Grupo Diédrico de un polígono de n lados*
V	*Grupo de Klein*
$\mathbb{C}_n$	*Grupo de las raíces n-ésimas de la unidad*
Q_2	*Grupo de los Cuaternios*
Q_n	*Generalización del grupo de los Cuaternios de orden $4n$*
$GL_n(\mathrm{F})$	*Grupo lineal general de grado n sobre el cuerpo* F
$H \leq G$	*H es subgrupo de G*
$H \trianglelefteq G$	*H es subgrupo normal de G*
G/H	*Grupo Cociente de G sobre H*
$[G : H]$	*Índice del subgrupo H en G*
$G \cong G'$	*G es isomorfo a G′*
$Z(G)$	*Centro de G*
$N_G(A)$	*Normalizador de A en G*
$\ker(\varphi), \mathrm{im}(\varphi)$	*Núcleo e imagen del homomorfismo φ*
$\langle A\rangle$, $\langle x\rangle$	*Grupo generado por el conjunto A; por el elemento x*
$G = \langle \ldots \mid \ldots \rangle$	*Generadores y relaciones (presentación) del grupo G*
$\mathrm{Aut}(G)$	*Grupo de Automorfismos de G*
$\mathrm{Hom}(X, Y)$	*Conjunto de homomorfismos de X a Y*
$H \times K$	*Producto directo de H y K*
$H \rtimes K$	*Producto semidirecto de H y K*
$a \mid b$	*a es divisor de b*
$a \equiv b \, mod(c)$	*a es congruente con b módulo c*
n_p	*Número de p-subgrupos de Sylow*

Parte I

PARTE MATEMÁTICA

. . . algunos misterios siempre escaparán a la mente humana. Para convencernos de ello, sólo hay que echar un vistazo a las tablas de los números primos, y ver que no reina ni orden, ni reglas . . . (Galois)

1

INTRODUCCIÓN A LA TEORÍA DE GRUPOS

1.1 INTRODUCCIÓN HISTÓRICA

Hoy en día, la Teoría de Grupos es una rama de las matemáticas plenamente consolidada. Sin embargo, esta teoría surge por la abtracción de muchas ideas que se han ido desarrollando paralelamente durante el transcurso de los siglos.

A finales del siglo XVIII, Lagrange (1736, 1813) estudió diferentes métodos para la resolución de ecuaciones polinómicas de grado tres y cuatro. Más tarde, Ruffini (1765-1822) afirmó que a partir de la ecuación quinta, las ecuaciones polinómicas no son resolubles por radicales. Esta demostración no se realizó hasta 1824 por Abel (1802,1829) y hoy en día se conoce como *Teorema de Abel-Ruffini*. En términos modernos esto quiere decir que A_n es simple y S_n no es resoluble para $n \geq 5$. Otra contribución destacable de Ruffini fue el estudio del grupo de Permutaciones, que más adelante usó Cauchy (1789,1857) para desarrollarla y dar lugar a la Teoría de Grupos de Permutaciones.

Todos estos estudios fueron precursores y contribuyeron en gran medida al trabajo realizado por Galois (1811-1832), un joven francés que revolucionó el mundo de las Matemáticas y, en particular, el concepto de «*Grupo*». A la temprana edad de 17 años, Galois escribió uno de sus primeros artículos, que se basaba en dar criterios sobre la resolubilidad de la ecuación polinómica por radicales. Además, escribió otros artículos dirigidos a la Academia de Ciencias Francesa, siendo todos rechazados por ser "incomprensibles y carecer de rigor". Entre sus estudios, agrupados en lo que se conoce como la *Teoría de Galois* o *Teoría de Grupos* y recogidas en *Oeuvres mathématiques d'Évariste Galois*, se destaca el grupo de Permutaciones o el grupo de Galois de un polinomio.

En 1851, Betti (1823,1892) relacionó la Teoría de Permutaciones y Teoría de Ecuaciones. Además, consigue demostrar que el grupo de Galois asociado a una ecuación es, en el sentido moderno, un grupo de permutaciones. A raíz de su trabajo, en 1870, Jordan (1838,1922) define el concepto de isomorfismo entre ellos. Además, prueba lo que hoy se conoce como el *Teorema de Jordan-Hölder*, que garantiza la existencia y unicidad, salvo isomorfismos, de series de composición para estos grupos.

Debemos destacar al matemático Walter von Dyck, (1856, 1934) quien, con la ayuda de Klein (1849,1925) dio una construcción de grupos libres y una definición de gru-

po dado por generadores y relatores, agrupados bajo el nombre de presentación de grupo.

En 1854, Cayley (1821,1895) define el concepto de grupo abstracto mediante una tabla de multiplicación que refleja los elementos. Posteriormente, publica varios artículos en los que demuestra que los cuaternios y las matrices forman un grupo. Además, demuestra el teorema que hoy en día lleva su nombre, el *Teorema de Cayley*, que sostiene que todo grupo puede describirse en términos de permutaciones.

A partir de este momento, fueron muchos los matemáticos quienes contribuyeron a construir la Teoría de Grupos que hoy en día conocemos. Podríamos destacar las investigaciones realizadas sobre grupos y subgrupos de orden primo. Los estudios de Cayley motivaron a Hölder (1859,1937), y en 1893 comenzó a investigar grupos cuyo orden es producto de números primos. Cabe destacar también a Sylow (1832,1918), quien enunció y demostró los teoremas que hoy en día llevan su nombre, y que nos proporcionan información sobre el número de subgrupos de orden fijo contenidos en un grupo finito dado.

1.2 OBJETIVOS

El principal objetivo del presente trabajo es el de desarrollar el trabajo realizado por Dyck, estudiando la construcción de grupos libres y principales problemas que surgen a raíz de éstos. Esta construcción permitirá definir grupos mediante un conjunto de generadores y relatores, que se usará en 3.2 (junto a las acciones de grupo) para introducir el producto semidirecto de grupos. Además, se estudiará principalmente la clasificación de grupos cuyo orden es producto de números primos, al igual que hizo Hölder.

En relación con la parte informática, se implementarán los principales grupos y se completarán y añadirán a la librería mencionada anteriormente nuevos métodos relacionados con la parte matemática. Uno de los problemas que surgen al dar un grupo mediante una presentación, es determinar si dos palabras definen el mismo elemento, o equivalentemente, cuándo una palabra define el elemento neutro. Este problema se conoce como Problema de Palabras, y para resolverlo se llevará a cabo una descripción e implementación del *Algoritmo de Todd Coxeter*, lo que permitirá definir grupos dados mediante una presentación. Asimismo, se deberán usar los métodos de la librería para identificar el grupo finitamente presentado con uno conocido.

La principales fuentes de bibliografía consultadas para la parte matemática han sido: [DF03], [Jud17], [Mil20], [BHLY15] y [BM13], y para informática: [DBO06], [HR94], [Mil16] y [GSBM16].

1.3 CONCEPTOS PREVIOS

A modo introductorio, definiremos en esta primera sección el concepto de grupo y algunas propiedades elementales sobre éstos, que nos facilitarán el entendimiento de los principales teoremas de la Teoría de Grupos utilizados en el desarrollo del trabajo. Seguiremos en 1.3.1 explicando las acciones de grupo, y en 1.3.2 enunciando los *Teoremas de Sylow*, que nos serán de mucha utilidad en las secciones posteriores. No se incluyen las demostraciones por ser un material clásico presente en cualquier curso sobre Teoría de Grupos. El lector puede consultar [Bue15] o [Mil20] para una descripción más detallada.

Definición 1. Un grupo es una cuaterna $(G, \bullet, ^{-1}, 1)$, donde G es un conjunto no vacío, $\bullet : G \times G \to G$ es una operación binaria, $^{-1} : G \to G$ es una operación unaria y $1 \in G$ es un elemento. Tales que se cumplen las siguientes propiedades:

1. *Asociatividad*: $x \bullet (y \bullet z) = (x \bullet y) \bullet z$, para todo $x, y, z \in G$.
2. *Existencia de elemento neutro*: $x \bullet 1 = 1 \bullet x$, para todo $x \in G$.
3. *Existencia de elemento simétrico*: $x \bullet x^{-1} = x^{-1} \bullet x = 1$, para todo $x \in G$.

Adicionalmente, si la operación binaria sobre G cumple la propiedad conmutativa, esto es: $x \bullet y = y \bullet x$, para todo $x, y \in G$, diremos que el grupo es *abeliano* o *conmutativo*.

Definición 2. Sea H un subconjunto no vacío de un grupo G. H es un subgrupo de G si verifica las siguientes propiedades:

1. Si $x, y \in H$, entonces $xy \in H$,
2. Si $x \in H$, entonces $x^{-1} \in H$.

Definición 3. Un grupo G es finito si contiene un número finito de elementos. A este número lo denominaremos orden del grupo G y se denotará por $|G|$.

Definición 4. El orden de un elemento $x \in G$, denotado por $|x|$, es el menor entero positivo n que cumple $x^n = 1$. Si no existe tal n, se dice que x tiene un orden infinito.

Mostramos diferentes grupos que se usarán con posterioridad:

Ejemplo 1.

- $\mathbb{Z}_n$, $n \in \mathbb{N}$, *el conjunto de las clases de equivalencia de los enteros* $0, 1, \ldots, n-1$ *módulo* n.
- *Sea* X *un conjunto no vacío. El conjunto de todas las aplicaciones biyectivas de* X *en sí mismo forman un grupo bajo la composición de funciones, que será denotado por* S_X. *Si* X *es el conjunto finito* $\{1, 2, \ldots n\}$, *entonces el grupo* S_X *se denotará por* S_n *y se llamará grupo Simétrico.*
- *El conjunto de las permutaciones pares del conjunto* $\{1, 2, \ldots, n\}$ *es un subgrupo de* S_n *que se conoce como grupo Alternado, denotado por* A_n.

- *Sea P un polígono regular de n lados. El grupo Diédrico D_n vendrá dado por las isometrías que preservan dicho polígono: las n rotaciones y n reflexiones.*

$$D_n = \{\underbrace{1, \varphi, \varphi^2, ..., \varphi^{n-1}}_{n \text{ rotaciones}}, \underbrace{\sigma, \sigma\varphi, ..., \sigma\varphi^{n-1}}_{n \text{ simetrías}}\}.$$

- *Sea n un número natural. Consideremos el conjunto de los números complejos que cumplen $z^n = 1$:*

$$\mathbb{C}_n = \{z \in \mathbb{C} : z^n = 1\}.$$

El conjunto $\mathbb{C}_n$ bajo la multiplicación usual de los números complejos se conoce como grupo de las raíces n-ésimas de la unidad, donde cada raíz compleja viene dada por:

$$\zeta_k = e^{\frac{2\pi ki}{n}} = cos(\frac{2\pi k}{n}) + isen(\frac{2\pi k}{n}) \ \ con\ k = 0, \ldots, n-1.$$

- *Los cuaternios son una extensión de los números reales, similar a los números complejos, donde se usan tres unidades imaginarias i, j, k que verifican:*

$$i^2 = j^2 = k^2 = ijk = -1\,.$$

El conjunto $\{\pm 1, \pm i, \pm j, \pm k\}$ se denomina grupo de los Cuaternios y se denota por Q_2.

El siguiente teorema será de gran importancia en la parte de informática ya que podremos representar los elementos de un grupo como permutaciones.

Teorema 1 (Cayley)**.** *Todo grupo es isomorfo a un subgrupo de un grupo de Permutaciones.*

Teorema 2. *Sea G un grupo y x un elemento de G. El subgrupo cíclico generado por x se define como $\langle x \rangle = \{x^n,\ n \in \mathbb{Z}\}$. Cuando todo el grupo G es generado por alguno de sus elementos entonces G es cíclico y se denotará $G = \langle x \rangle$.*

Definición 5. Sea H un subgrupo de G. Se define la clase lateral a izquierda de un elemento $x \in G$ respecto de H como:

$$xH = \{xh;\ h \in H\}.$$

De igual modo, la clase lateral a derecha de x respecto a H viene dada por:

$$Hx = \{hx;\ h \in H\}.$$

El número de clases laterales izquierda coincide con el número de clases laterales derechas, se denomina índice de H en G y se denota por $[G : H]$.

A continuación enunciamos el *Teorema de Lagrange*, uno de los teoremas más importantes de la Teoría de Grupos ya que tiene implicaciones muy importantes en el estudio de grupos finitos.

Teorema 3 (Lagrange)**.** *Sea H un subgrupo de un grupo finito G. Entonces, el orden de H divide al orden de G. En particular, se cumple:*

$$|G| = [G : H] \cdot |H| .$$

Definición 6. Un subgrupo H de un grupo G se dice que es un subgrupo normal de G y se denotará $H \trianglelefteq G$ si $xH = Hx$, para todo $x \in G$.

El siguiente teorema servirá para demostrar la normalidad de un subgrupo.

Teorema 4. *Un subgrupo H de un grupo G es normal si, y sólo si, $xHx^{-1} \subseteq H$, para todo $x \in G$.*

Los subgrupos normales tienen especial importancia. Cuando un subgrupo H es normal en G, entonces el conjunto de las clases laterales a izquierda y a derecha coinciden; de hecho, forman un grupo llamado Grupo Cociente, que será denotado por G/H. Enunciamos el siguiente teorema:

Teorema 5. *Sea H un subgrupo normal de un grupo G. El conjunto $G/H = \{Hx,\ x \in G\}$ es un grupo con la operación $(Hx)(Hy) = Hxy$.*

Terminamos este primer capítulo introductorio recordando el concepto de homomorfismo de grupos.

Definición 7. Sean G y G' dos grupos. Un homomorfismo de grupos de G en G' es una aplicación $\varphi\colon G \to G'$ que satisface $\varphi(x, y) = \varphi(x)\varphi(y)$, para todo $x, y \in G$.

Definición 8. Un isomorfismo es un homomorfismo de grupos biyectivo. Dos grupos G y G' serán isomorfos y se denotará por $G \cong G'$ si existe un isomorfismo entre ellos. Un isomorfismo de un grupo G en sí mismo es llamado automorfismo de G y se denota por $\mathrm{Aut}(G)$ al conjunto de todos los automorfismo de G.

1.3.1 *Acciones de grupo*

En primer lugar, se estudiará el caso de grupos actuando sobre conjuntos, idea que nos permitirá obtener más información sobre la estructura del grupo. Después veremos la teoría necesaria para hacer actuar a un grupo sobre otro.

Definición 9. Sea X un conjunto y G un grupo. Una acción (izquierda) de G sobre X es una aplicación $G \times X \to X$; $(g, x) \mapsto {}^g x$ que cumple las propiedades:

1. ${}^1 x = x, \quad \forall x \in X.$
2. ${}^g({}^{g'} x) = {}^{gg'} x, \quad \forall x \in X, \forall g, g' \in G.$

En tal caso se dice que G actúa sobre X, y que el conjunto X es un G-conjunto.

De forma análoga, se pueden definir acciones a derechas:

$$\begin{aligned} G \times X &\to X \\ (g, x) &\mapsto x^g \end{aligned}$$

Estas nos serán de mucha utilidad más adelante en la Sección 2.3.1, cuando se proceda a describir el *Algoritmo de Todd Coxeter*.

Comentario. Sea G un grupo y X un conjunto no vacío. Dar una acción de G sobre X es equivalente a dar un homomorfismo de grupos $\phi\colon G \to S(X)$, el grupo de Permutaciones de X.

$$\begin{aligned} \phi\colon G &\longrightarrow S(X) \\ g &\longmapsto \phi(g) = \phi_g\colon \quad X \longrightarrow X \\ &\qquad\qquad\quad x \longmapsto \phi_g(x) = {}^g x \end{aligned}$$

Definición 10. El *núcleo* de una acción $\phi\colon G \times X \to X$ es el conjunto de los elementos de G que actúan trivialmente sobre todo elemento del conjunto X:

$$\ker(\phi) = \{g \in G \mid {}^g x = x,\ \forall x \in X\}\,.$$

Cuando ϕ es inyectivo, o equivalentemente el núcleo es trivial, decimos que la acción es *fiel*.

En cambio, los elementos del conjunto X sobre los que todos los elementos de G actúan trivialmente son llamados *puntos fijos*:

$$\mathrm{Fix}(X) = \{x \in X \mid {}^g x = x,\ \forall g \in G\}\,.$$

Ahora bien, en vez de considerar un conjunto X, podemos tomar otro grupo H, con $h, h' \in H$ y restringirnos a las acciones de G sobre H que sean compatibles con la estructura de grupo, es decir, acciones que satisfagan además:

1. ${}^g 1 = 1$,
2. ${}^g(hh') = {}^g h\, {}^g h'$.

Bajo estas condiciones, el grupo H se denomina G-grupo.

Entonces, dar una acción de grupos de G en H equivale, por el Comentario 1.3.1, a dar un homomorfismo de grupos $G \to \mathrm{Aut}(H)$. Este concepto nos será de mucha utilidad en la sección 3.2 ya que dará lugar a la construcción del producto semidirecto. Tenemos pues:

$$\begin{aligned} G \times H &\to G \\ (g,h) &\mapsto {}^{g}h \end{aligned}$$

o, equivalentemente el homomorfismo de grupos:

$$\begin{aligned} \phi\colon G &\longrightarrow \mathrm{Aut}(H) \\ g &\longmapsto \phi(g) = \phi_g\colon \quad H \longrightarrow H \\ & \qquad\qquad\quad h \longmapsto \phi_g(h) = {}^{g}h \end{aligned}$$

Ejemplo 2. *Sea G un grupo actuando sobre sí mismo. La acción por traslación se define como:*

$$\begin{aligned} G \times G &\to G \\ (g,x) &\mapsto {}^{g}x := gx \end{aligned}$$

Claramente satisface la Definición 9 ya que:

1. ${}^{1}x = 1x = x$,
2. ${}^{gg'}x = (gg')x = g(g'x) = {}^{g}({}^{g'}x)$.

Sin embargo, no es una acción de grupos ya que ${}^{g}1 = g \cdot 1 = g \neq 1$.

Ejemplo 3. *De igual modo que en 2, la acción por conjugación se define como:*

$$\begin{aligned} G \times G &\to G \\ (g,x) &\mapsto {}^{g}x := gxg^{-1} \end{aligned}$$

Satisface la Definición 9 pues:

1. ${}^{1}x = 1x1^{-1} = x$,
2. ${}^{gg'}x = (gg')x(gg')^{-1} = g(g'xg'^{-1})g = {}^{g}({}^{g'}x)$.

Cumple 1 y 2 por lo que además es una acción de grupo.

1.3.2 *Teoremas de Sylow*

Los *Teoremas de Sylow* son nombrados en honor al matemático Ludwig Sylow y son de los teoremas más importantes en Teoría de Grupos ya que nos proporcionan información detallada sobre los subgrupos de orden p, donde p es un número primo. Entre sus aplicaciones destacamos su importancia en la clasificación de grupos no abelianos finitos.

Definición 11. Si p es un primo, un grupo G es un p-grupo si todo elemento tiene orden una potencia de p.

El Teorema 6 nos permitirá probar que todo p-grupo finito tienen orden una potencia de p:

Teorema 6 (Cauchy)**.** *Sea G un grupo y p un número primo tal que p divide a $|G|$, entonces G contiene un elemento de orden p.*

Comentario. Dado G un grupo de orden n y p primo que divida al orden de G, el Teorema de Cauchy (6) nos asegura la existencia de un subgrupo de orden p. Como consecuencia, el orden de un $p-$grupo de orden finito es una potencia de p.

Sea G un grupo finito con $|G| = n$ y p^r la mayor potencia de p que divide a n. Un subgrupo de G de orden p^r es llamado p-subgrupo de Sylow de G. Podemos enunciar ahora los *Teoremas de Sylow*, agrupados en el Teorema 7.

Teorema 7 (Teoremas de Sylow)**.** *Sea G un grupo finito de orden n y p un primo que divide a n, donde $n = p^r m$, con p y m primos relativos. Entonces:*

1. *(Sylow I) Para cada primo p que divida a n, existe un p-subgrupo de Sylow.*
2. *(Sylow II) Si H, K son p-subgrupos de Sylow de G, entonces H y K son subgrupos conjugados de G, es decir, $H = gKg^{-1}$ para algún $g \in G$.*
3. *(Sylow III) Denotando n_p como el número de p-subgrupos de Sylow, se tiene que $n_p \equiv 1\ mod(p)$ y n_p divide a m.*

Comentario. Una consecuencia útil del Teorema de Sylow III 3 es que $n_p = 1$ equivale a decir que el único p-subgrupo de Sylow ha de ser un subgrupo normal.

2

PRESENTACIONES DE GRUPOS

Actualmente, un *grupo abstracto* debe satisfacer la Definición 1. Sin embargo, podemos describir el grupo y sus elementos de dos formas:

1. Enumerando los elementos del grupo junto a la operación binaria, de igual modo que se han dado en el Ejemplo 1 anterior.
2. En términos de generadores y relatores. En secciones anteriores se han introducido grupos como el grupo Diédrico D_n o el grupo de los Cuaternios Q_2, cuyos elementos satisfacen una serie de propiedades.

   El grupo D_n está generado por φ y σ, que cumplen:

   $$\varphi^n = 1, \quad \sigma^n = 1, \quad \sigma\varphi = \varphi^{-1}\sigma\,.$$

   En el caso de Q_2, podemos generalizar el grupo y obtener el grupo generalizado de los Cuaternios Q_n, de forma que sus elementos cumplan las siguientes relaciones:

   $$x^{2n} = 1, \quad x^n = y^2 \quad e \quad yxy^{-1} = x^{-1}\,.$$

   Por ello, podemos pensar en presentar un grupo como un conjunto de generadores S de G y un conjunto de relatores R que los elementos de S deben satisfacer para determinar G. Esto es lo que se conoce como *presentación de un grupo* y se precisará en esta sección. Para llevar a cabo esto, necesitamos introducir primero el concepto de *grupo libre*. La documentación usada para esta sección ha sido principalmente [BHLY15] e [IC99].

2.1 GRUPO LIBRE

Sea X un conjunto arbitrario. Consideramos el conjunto $X^{\pm 1} = X^{+1} \cup X^{-1}$ donde cada elemento $x \in X$ tendrá un elemento $x^{+1} \in X^+$ y otro asociado $x^{-1} \in X^{-1}$. Para simplificar notación, identificamos X^{+1} con X.

Una *palabra* en X es una secuencia finita de elementos

$$w = x_{a_1}^{\epsilon_1} x_{a_2}^{\epsilon_2} \cdots x_{a_n}^{\epsilon_n} \quad \text{con } x_{a_i} \in X,\ \epsilon_i \in \{+1, -1\},\ n \in \mathbb{N}; \tag{1}$$

y será *reducida* si no contiene subpalabras del tipo xx^{-1} o $x^{-1}x$. Por otro lado, su longitud estará determinada por el número de elementos que contiene, $|w| = n$. Asumimos que la palabra vacía, ϵ, es reducida, luego $|\epsilon| = 0$.

Si G es un grupo y $X \subseteq G$, una palabra de G en X es una expresión w como en (1) en la que yuxtaposición indica producto y $^{-1}$ indica inverso.

Definición 12. Un subconjunto X de un grupo G es un sistema de generadores si todo elemento de G se puede escribir como una palabra (reducida) en X.

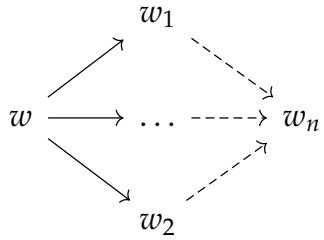
Comentario. Todo grupo tiene un sistema de generadores, basta con tomar el propio grupo como conjunto.

Definición 13. Diremos que los elementos de un subconjunto $X \subseteq G$ son independientes si la única palabra reducida en X que es igual a $1 \in G$ es la palabra vacía.

Definición 14. Un grupo G es libre si tiene una base, esto es, si existe un subconjunto X de elementos independientes que son un sistema de generadores, donde cada palabra reducida no vacía en $X^{\pm 1}$ define un elemento no trivial en G. Llamamos rango de un grupo libre al cardinal de cualquiera de sus bases.

Sea X un conjunto arbitrario. Nuestro objetivo ahora es la *"construcción"* de un grupo libre con base X. Para ello, se seguirá un proceso de reducción sobre el conjunto de palabras que contiene, en el que se eliminarán subpalabras del tipo xx^{-1} y $x^{-1}x$ hasta obtener un conjunto en el que todas las palabras sean reducidas. Se deberán tener en cuenta las siguientes consideraciones.

1. **El proceso de reducción no es único**: En la clase de equivalencia de una palabra existe una única palabra reducida, aunque puede haber distintos procesos de reducción para obtenerla.



2. **Palabras equivalentes**: Diremos que dos palabras son equivalentes si podemos pasar de una a otra por un proceso de reducción.

Dado un conjunto X, denotamos por $\mathrm{F}(X)$ al conjunto de sus palabras reducidas.

A partir de dos palabras reducidas, w_1 y w_2, podemos definir su **producto** como la única palabra reducida en la clase de la palabra que se obtiene por concatenación de ambas:

$$\left.\begin{aligned} w_1 &= x_{a_1}^{\epsilon_1} \cdots x_{a_n}^{\epsilon_n} \\ w_2 &= x_{b_1}^{\epsilon_1} \cdots x_{b_m}^{\epsilon_n} \end{aligned}\right\}$$

Definimos $w_1 \cdot w_2$ como la única palabra reducida en la clase de $x_{a_1}^{\epsilon_1} \cdots x_{a_n}^{\epsilon_n} x_{b_1}^{\epsilon_1} \cdots x_{b_m}^{\epsilon_m}$; es decir, sobre la concatenación $w_1 w_2$ se realiza un proceso de reducción eliminando parejas de elementos asociados si fuera necesario.

Teorema 8 (Existencia)**.** *El conjunto* $\mathrm{F}(X)$ *de palabras reducidas en* X *dotadas con el producto anterior forman un grupo libre con base el conjunto* X*.*

Demostración. La operación está bien definida ya que el producto de dos palabras reducidas dotado con el proceso de reducción da lugar a una palabra reducida.

- La palabra vacía ϵ es la identidad.

$$w = \epsilon \cdot w = w \cdot \epsilon \,.$$

- El elemento inverso de una palabra $w = x_{a_1}^{\epsilon_1} x_{a_2}^{\epsilon_2} \cdots x_{a_r}^{\epsilon_r}$ viene dado por:

$$w^{-1} = x_{a_r}^{-\epsilon_r} \cdots x_{a_2}^{-\epsilon_2} x_{a_1}^{-\epsilon_1}.$$

- Veamos que se cumple también la propiedad asociativa, es decir:

$$w_1(w_2 w_3) = (w_1 w_2) w_3 \,.$$

  Realicemos un proceso inductivo sobre la longitud de w_2.

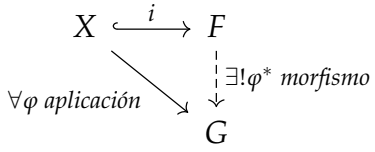
  1. $|w_2| = 1$ (se tiene que $w_2 = x$ o $w_2 = x^{-1}$, para algún $x \in X$). A partir de esto nos encontramos con las siguientes situaciones: Que el último elemento de w_1 sea o no asociado con w_2 y que el primer elemento de w_3 sea o no asociado con w_2. En las 4 posibilidades se ve claramente que $w_1(w_2 w_3) = (w_1 w_2) w_3$.
  2. Supuesto cierto para $|w_2| \leq n$, veamos que se cumple para $|w_2| = n + 1$.

     Sea $w_2 = w_k x^{\epsilon}$, donde $\epsilon = \pm 1$ y $x \in X$.

$$\begin{aligned} w_1(w_2 w_3) =& w_1((w_k x^{\epsilon}) w_3) = (w_1 w_k)(x^{\epsilon} w_3) \\ =& ((w_1 w_k) x^{\epsilon}) w_3 = (w_1 (w_k x^{\epsilon})) w_3 = (w_1 w_2) w_3 \,. \end{aligned}$$

□

Teorema 9 (Propiedad Universal del grupo libre)**.** *Sea* F *un grupo libre y* $X \subseteq \mathrm{F}$ *una base de* F*. Entonces, para cualquier grupo* G*, dar un morfismo de grupos* $\varphi^* : F \to G$ *es equivalente a dar una aplicación* $\varphi : X \to G$*. En otras palabras:*



Demostración. Como X es base de F, se tiene que cada elemento $w \in \mathrm{F}$ se define por una única palabra reducida en $X^{\pm 1}$,

$$w = x_{a_1}^{\epsilon_1} \cdots x_{a_n}^{\epsilon_n} \quad \text{con } x_{a_i} \in X,\ \ \epsilon_i \in \{+1, -1\}, n \in \mathbb{N}.$$

Dado φ, definimos φ^* como:

$$\varphi^*(w) = \varphi^*(x_{a_1}^{\epsilon_1} \cdots x_{a_n}^{\epsilon_n}) = \varphi(x_{a_1})^{\epsilon_1} \cdots \varphi(x_{a_n})^{\epsilon_n}. \tag{2}$$

donde se ve que claramente φ^* es un homomorfismo de grupos y el diagrama conmuta.

Cualquier homomorfismo $\varphi^* \colon \mathrm{F} \to G$ que haga el diagrama conmutativo debe satisfacer (2), por tanto, φ^* es único. □

Teorema 10 (Unicidad)**.** *Si G es un grupo libre y X es una base de G, entonces G es isomorfo a $\mathrm{F}(X)$.*

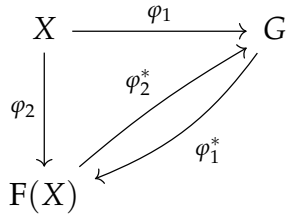
Demostración. Definimos las inclusiones:

$$X \xrightarrow{\varphi_1} G\,; \quad X \xrightarrow{\varphi_2} F(X).$$

Por la propiedad universal de grupos libres, como X es base de G y de $\mathrm{F}(X)$, existirán dos homomorfismos:

$$\varphi_1^* \colon G \to \mathrm{F}(X) \quad \text{tal que} \quad \varphi_1^* \circ \varphi_1 = \varphi_2, \tag{3}$$
$$\varphi_2^* \colon \mathrm{F}(X) \to G \quad \text{tal que} \quad \varphi_2^* \circ \varphi_2 = \varphi_1. \tag{4}$$

En otras palabras:



La composición $(\varphi_1^* \circ \varphi_2^*) \colon \mathrm{F}(X) \to \mathrm{F}(X)$ es un homomorfismo que cumple:

$$(\varphi_1^* \circ \varphi_2^*) \circ \varphi_2 = \varphi_1^* \circ (\varphi_2^* \circ \varphi_2) = \varphi_1^* \circ \varphi_1 \overset{(3)}{=} \varphi_2$$

Análogamente, se tiene que $(\varphi_2^* \circ \varphi_1^*) \circ \varphi_1 = \varphi_1$. Por tanto, $(\varphi_1^* \circ \varphi_2^*)$ y $(\varphi_2^* \circ \varphi_1^*)$ son, respectivamente, las aplicaciones identidad en $\mathrm{F}(X)$ y G, obteniendo así que $G \cong \mathrm{F}(X)$. □

Teorema 11. *Si X e Y son dos bases de un grupo libre* F, *entonces* $|X| = |Y|$.

Demostración. Por la propiedad universal del grupo libre, cualquier aplicación $X \to \mathbb{Z}_2$ da lugar a un homomorfismo de F en el grupo cíclico $\mathbb{Z}_2$. Denotando por $\mathrm{Hom}(\mathrm{F}, \mathbb{Z}_2)$ al conjunto de homomorfismos de F a $\mathbb{Z}_2$, se tiene que $|\mathrm{Hom}(\mathrm{F}, \mathbb{Z}_2)| = 2^{|X|}$. Esto implica

$$2^{|X|} = 2^{|Y|}\text{, por lo que } |X| = |Y|.$$

□

Teorema 12. *Dos grupos libres* $\mathrm{F}(X)$ *y* $\mathrm{F}(Y)$ *son isomorfos si, y sólo si,* $|X| = |Y|$.

Demostración.

Suficiencia. Supongamos que $|X| = |Y|$. Como ambos conjuntos tienen la misma cardinalidad, existe una correspondencia uno a uno, a la que llamamos f:

$$f\colon X \to Y \quad y \quad f^{-1}\colon Y \to X.$$

Como $\mathrm{F}(X)$ y $\mathrm{F}(Y)$ son grupos libres, el teorema universal de grupos libres nos garantiza la existencia de únicos homomorfismos φ y φ^{-1} que extienden a f y f^{-1}. Por tanto:

$$\varphi\colon \mathrm{F}(X) \to \mathrm{F}(Y) \quad y \quad \varphi^{-1}\colon \mathrm{F}(Y) \to \mathrm{F}(X).$$

La composición $(\varphi \circ \varphi^{-1})\colon \mathrm{F}(X) \to \mathrm{F}(X)$ extiende la identidad en X. De igual forma, $(\varphi^{-1} \circ \varphi)\colon \mathrm{F}(Y) \to \mathrm{F}(Y)$ extiende la identidad en Y. Así, $\mathrm{F}(X) \cong \mathrm{F}(Y)$.

Necesidad. Supongamos ahora que $\mathrm{F}(X) \cong \mathrm{F}(Y)$. Consideremos el conjunto de homomorfismos de $\mathrm{F}(X)$ a $\mathbb{Z}_2$ y de $\mathrm{F}(Y)$ a $\mathbb{Z}_2$, denotados por $\mathrm{Hom}(\mathrm{F}(X), \mathbb{Z}_2)$ y $\mathrm{Hom}(\mathrm{F}(Y), \mathbb{Z}_2)$, respectivamente. Como $\mathrm{F}(X) \cong \mathrm{F}(Y)$, se tendrá que:

$$|\mathrm{Hom}(\mathrm{F}(X), \mathbb{Z}_2)| = |\mathrm{Hom}(\mathrm{F}(Y), \mathbb{Z}_2)|$$

y, por definición de grupo libre, habrá tantos homomorfismos de $\mathrm{F}(X)$ en $\mathbb{Z}_2$ como aplicaciones de X en $\mathbb{Z}_2$. Análogo para $\mathrm{F}(Y)$ e Y, luego:

$$2^{|X|} = |\mathrm{Hom}(\mathrm{F}(X), \mathbb{Z}_2)| = |\mathrm{Hom}(\mathrm{F}(Y), \mathbb{Z}_2)| = 2^{|Y|},$$

lo cual implica que $|X| = |Y|$. □

Como consecuencia del Teorema 12, salvo isomorfismo hay sólo un grupo libre de un rango dado. Como norma general, F_n representa **el** grupo libre de rango n, salvo isomorfismos.

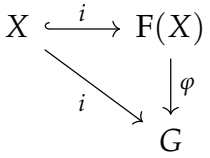
Teorema 13. *Todo grupo es isomorfo a un cociente de un grupo libre.*

Demostración. Sea G un grupo y X un sistema de generadores de G.

Consideremos $\mathrm{F}(X)$ el grupo libre sobre X, la inclusión $X \hookrightarrow G$ induce, por la propiedad universal del grupo libre, un homomorfismo:

$$\varphi\colon \mathrm{F}(X) \to G \text{ tal que } \varphi(x) = x, \text{ para todo } x \in X.$$

que hace conmutativo al diagrama:



El homomorfismo φ es sobreyectivo porque es la aplicación identidad en X, luego aplicando el primer Teorema de Isomorfía,

$$G = \mathrm{Im}(\varphi) \cong \mathrm{F}(X)/\ker(\varphi).$$

□

Teorema 14 (Nielsen-Schreier)**.** *Todo subgrupo de un grupo libre es libre. Además, si* F *es libre de rango n todo subgrupo suyo es libre de rango menor o igual a n.*

La demostración requiere de conceptos topológicos que no serán estudiados por lo que se puede consultar [Joh90] para una descripción detallada.

2.2 PRESENTACIÓN DE UN GRUPO

En el Teorema 13 se ha probado que todo grupo G es isomorfo a un cociente de un libre:

$$G \cong \mathrm{F}(X)/K,$$

donde $\varphi\colon \mathrm{F}(X) \twoheadrightarrow G$ es un epimorfismo y $K = ker(\varphi) \trianglelefteq G$, un subgrupo normal. Como X es un sistema de generadores de G, aplicando el Teorema 14, se tiene que K es libre y podremos tomar $R \subseteq K$ una base de K.

Los elementos de R son palabras en el alfabeto $X^{\pm 1}$, además, si $w \in R$ es la palabra $w = x_{a_1}^{\epsilon_1} \cdots x_{a_n}^{\epsilon_n}$, entonces:

$$\varphi^*(w) = \varphi(x_{a_1}^{\epsilon_1} \cdots x_{a_n}^{\epsilon_n}) = 1,$$

donde la yuxtaposición en la parte central de la igualdad es producto en G.

El par $\langle X \mid R \rangle$ se conoce como presentación de G. A los elementos de X los llamaremos generadores de G y los elementos de R son llamados relaciones o relatores.

Definición 15. Un grupo G se dice *finitamente generado* si admite un conjunto de generadores finito y *finitamente presentado* si tiene al menos una presentación finita, es decir, puede ser dado por un número finito de generadores y relatores. Naturalmente, la propiedad ser finitamente presentado implica ser finitamente generado.

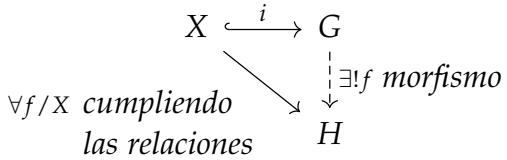
La notación usual para representar los grupos finitamente presentados es la siguiente:

$$G = \langle X \mid R \rangle = \langle x_1, \ldots, x_n \mid w_1, \ldots, w_m \rangle. \tag{5}$$

donde los elementos w_i , $i \in \{1, \ldots, m\}$ son palabras en $X^{\pm 1}$ y serán triviales cuando representan elementos de G.

Aunque en una presentación de un grupo G como (5), hemos dicho que el conjunto de relatores R es una base para el núcleo K, a veces sólo basta con exigir que R genere K; esto es, se suelen admitir presentaciones en las que los relatores no son necesariamente independientes.

Teorema 15 (Dyck). *Sea $G = \langle X \mid R \rangle$ un grupo definido por generadores X y relaciones R, y sea H un grupo cualquiera. Dar un morfismo $f : G \to H$ es equivalente a dar una aplicación $f : X \to H$ tal que los elementos de $f_*(X) \subseteq H$ cumplan las relaciones de R.*



Además, si $f_*(X)$ genera H, entonces f es un epimorfismo.

Demostración. Este teorema es una consecuencia inmediata de la definición de presentación de un grupo, y de las propiedades universales del grupo libre y el grupo cociente. □

Uno de los problemas que surgen al dar un grupo mediante una presentación es el de determinar cuando dos elementos del grupo (dados como palabras en los generadores) son iguales; es decir: determinar, usando las relaciones, si dos palabras en los generadores dan lugar al mismo elemento. Este problema es conocido con el nombre de Problema de Palabras (*Word Problem*) y es un problema abierto en la actualidad. Así, por ejemplo, no será fácil determinar a partir de una presentación del grupo, cuántos elementos tiene éste y cómo podemos escribirlos. El *Teorema de Dyck* 15 nos será de gran utilidad en este sentido.

En la actualidad, existen algoritmos que pueden ser capaces de resolver el Problema de Palabras para un grupo G definido por una presentación, como es el caso del *Algoritmo de Todd Coxeter*. Dado un subgrupo $H \leq G$, el algoritmo utiliza una técnica llamada enumeración de clases de G/H para obtener el índice $[G : H]$, y obteniendo una representación por permutaciones de G, se podrá identificar (mediante un isomorfismo) con un grupo conocido. Véase la sección 2.3.

Ejemplo 4. *Sea G un grupo que está dado por la presentación:*

$$G = \langle a \mid a^n \rangle.$$

Está claro que G está generado por un único elemento y, por tanto, G es un grupo cíclico. El generador a cumple la relación $a^n = 1$, lo que significa que su orden es un divisor de n, luego $|G| \leq n$. Este grupo se denotará C_n y tendrá n elementos. En efecto, si tomamos cualquier grupo cíclico de orden n, es obvio que su generador cumple la relación que define a G, por lo que aplicando el Teorema de Dyck 15:

$$\varphi \colon G \to C_n$$

es un epimorfismo y se tiene que $n = |C_n| \leq |G|$. Al tener la igualdad probada podemos afirmar que φ es isomorfismo.

Ejemplo 5. *Sea G un grupo definido por la siguiente presentación:*

$$G = \langle x, y \mid x^n,\ y^2,\ (xy)^2 \rangle.$$

Veamos que $G \cong D_n$. Consideremos $\varphi, \sigma \in D_n$, que verifican las relaciones:

$$\varphi^n = 1,\ \sigma^2 = 1,\ (\sigma\varphi)^2 = 1.$$

Aplicando el Teorema de Dyck 15, existe un epimorfismo tal que $\psi \colon G \to D_n$. Por otro lado, las relaciones de G nos indican que sus elementos deben ser de la forma:

$$x^k y^l\,, \quad 0 \leq k < n,\ 0 \leq l \leq 1\,,$$

que nos dice que $|G| = 2n = |D_n|$, lo que implica que ψ es un isomorfismo.

Ejemplo 6. *Consideramos el siguiente grupo dado por una presentación:*

$$G = \langle a, b \mid aba^{-1}b^{-1}b^{-1}, bab^{-1}a^{-1}a^{-1} \rangle.$$

Sabemos que el grupo tiene dos generadores, a y b, que deben satisfacer las relaciones R. En principio, no parece sencillo determinar los elementos que contiene el grupo, ni siquiera saber si el grupo es finito o no. Se podría realizar un proceso para identificar los elementos y ver como se opera entre ellos, sin embargo, sería muy costoso comprobar si dos palabras representan la misma (Problema de Palabras). A pesar de todos estos problemas, se puede aplicar el Algoritmo de Todd Coxeter para demostrar que este grupo es isomorfo al grupo trivial. Véase el Ejemplo 6.

2.3 PROBLEMA DE PALABRAS

Sea G un grupo definido por una presentación finita $\langle X \mid R \rangle$. Como se ha comentado anteriormente, el Problema de Palabras *(Word Problem)* para el grupo G cuestiona si existe un algoritmo para determinar si una palabra en $X^{\pm 1}$ representa el elemento identidad de G, o equivalentemente, determinar si dos palabras generan el mismo elemento. Fueron Nivikov y Boone quieres demostraron que se trataba de un problema indecidible y mostraron la existencia de grupos con presentación finita en el que no existía dicho algoritmo. En [Simo5] se presenta una prueba de este teorema.

Es importante destacar que aunque no haya un algoritmo general para resolver el Problema de Palabras dado un conjunto arbitrario de generadores y relatores, en muchos casos de grupos finitos se puede resolver mediante una técnica llamada enumeración de clases (*coset enumeration*) que explicaremos en la siguiente sección.

2.3.1 *Algoritmo de Todd Coxeter*

Dado un grupo G definido por una presentación y $H \leq G$ un subgrupo. La enumeración de clases es el problema de contar las clases de H en G. En 1936, J.A. Todd y H.S.M. Coxeter describieron el algoritmo original, que se caracterizaba por ser un método bastante mecánico enfocado para realizarse manualmente y que hoy en día lleva sus nombres, el *Algoritmo de Todd Coxeter*. No obstante, debido a su complejidad, se convirtió en uno de los primeros algoritmos del área de las Matemáticas en hacer uso de los ordenadores electrónicos cuando éstos estuvieron disponibles. El algoritmo original se describe en [TC36]; sin embargo, usaremos una explicación actualizada basada en [Mil16] y [DBO06] para desarrollarlo.

Separaremos la explicación del algoritmo en varias secciones. En 2.3.1.1, se detallará una descripción con diferentes ejemplos. En cambio, en 5.3 se dedicará una sección para el desarrollo informático de éste, donde se profundizará sobre la implementación y ejemplos de ejecución en Jupyter. Además se incorporará el Teorema 22, que probará que el algoritmo programado es correcto.

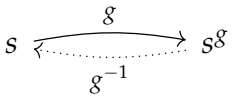
2.3.1.1 *Descripción del Algoritmo*

Consideramos G un grupo generado por un conjunto X y sea S un G-conjunto. Un *Grafo de Schreier* Γ es una grafo dirigido donde el conjunto de vértices es S y existe una arista dirigida de $s \to s^g$ para cada $g \in X$ y $s \in S$. Se usará notación exponencial y acciones a la derecha, ya que nos proporcionan una forma más sencilla de seguir el camino que sigue una palabra mientras la leemos de izquierda a derecha. s^g indica el elemento obtenido al actuar g sobre s.

$$s \xrightarrow{g} s^g$$

Los elementos $g \in X$ se denominan etiquetas, y para representar el grafo, hay que tener en cuenta las siguientes consideraciones:

- Dada una etiqueta $g \in X$, g denota el recorrido en un determinado sentido de la arista a partir de un vértice, mientras que g^{-1} denota el recorrido en sentido opuesto.



- Cada etiqueta será representada por un único color.

Sea G un grupo definido por una presentación $\langle X \mid R \rangle$, donde X es el conjunto de generadores y R el conjunto de relatores. Sea $H = \langle Y \rangle = \langle h_1, h_2, \ldots, h_r \rangle$ un subgrupo de G donde los generadores h_i son palabras del alfabeto $X^{\pm 1}$. El *Algoritmo de Todd Coxeter* obtiene el índice de H en G mediante la enumeración de clases (a derechas) de G/H.

Comentario (Notación). Cuando el subgrupo H es normal en G, el conjunto de las clases laterales izquierdas y derechas coinciden. Si no es normal, las clases no coinciden por lo que para simplificar notación, denotaremos por G/H al conjunto de clases laterales derechas. Véase la definición 5.

La idea que sigue el algoritmo es generar un grafo de Schreier que refleje la acción a la derecha de G sobre G/H. Las distintas clases serán denotadas por $1, 2, \ldots$ y se irán añadiendo nuevas si es necesario. El algoritmo termina cuando el grafo se completa, esto significa:

1. Dada una etiqueta $g \in X$, cada vértice tendrá exactamente una arista saliendo y otra entrando con dicha etiqueta.

2. Cada uno de los relatores de G se deben satisfacer en cada uno de los vértices; es decir, se debe poder realizar un recorrido marcado por cada relator que empiece y termine en cada vértice. Este proceso de verificación se conoce como escaneo (*scanning*).

A continuación, se describirá el algoritmo en términos de un grafo de Schreier.

1. Comenzamos con un único vértice que será etiquetado con el valor 1.
2. Para cada elemento de Y, y partiendo desde el vértice 1, se debe realizar un recorrido en el que se satisfaga cada una de las relaciones. En el proceso, se pueden definir nuevos vértices si es necesario. Cada vértice nuevo definido se debe etiquetar con el siguiente número natural disponible.
3. Para cada nuevo vértice definido:
   a) Etiquetar el vértice con el siguiente $n \in \mathbb{N}$ disponible.
   b) Para cada relación de R, se debe poder realizar un recorrido que empiece y acabe en este vértice. Si el recorrido no se puede llevar a cabo, se debe definir un nuevo vértice.

   Hay que tener en cuenta que el procedimiento únicamente acabará si G/H es finito. En caso contrario, se definirían infinitos vértices y el proceso se ejecutaría indefinidamente.

Ilustramos este procedimiento mediante el siguiente ejemplo:

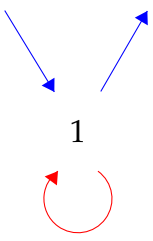
$$G := \langle X \mid R \rangle = \langle a, b \mid a^3, b^3, (ab)^2 \rangle.$$

Sea $H := \langle a \rangle \leq G$, el subgrupo de G generado por a, y consideremos la acción a la derecha de G sobre G/H. El conjunto de generadores X está formado por dos elementos; luego tomaremos dos colores: el rojo para representar la acción de a y azul para representar la acción de b.

Las distintas relaciones y recorridos que se deben verificar son los siguientes:

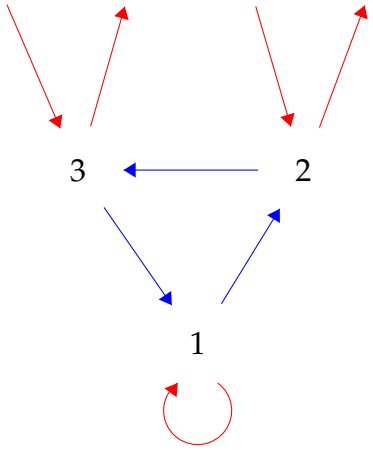
- $a^3 = 1$, recorrido: «rojo, rojo, rojo»,
- $b^3 = 1$, recorrido: «azul, azul, azul»,
- $(ab)^2 = 1$, recorrido: «rojo, azul, rojo, azul».

1. Definimos $1 := 1^a$ e inicializamos el grafo. Como consecuencia, habrá una arista de color rojo que empieza y termina en el vértice 1.

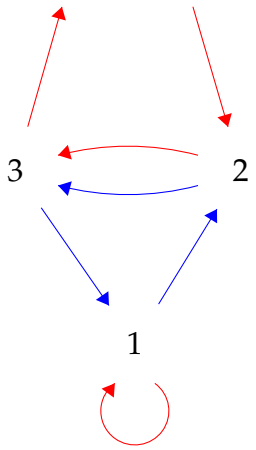


   Las aristas azules nos avisan de que el vértice debe tener una arista azul entrando y otra saliendo, sin embargo, no podemos asegurar que $1 = 1^b$.

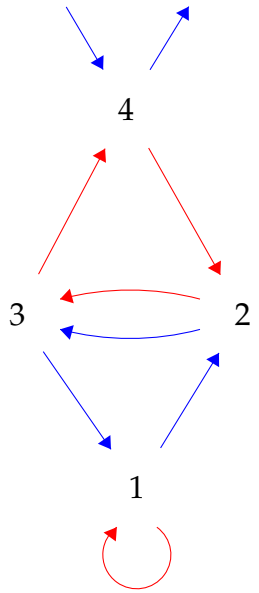
2. Vemos que el vértice 1 ya cumple la relación $a^3 = 1$. La segunda relación no se verifica, por lo que procedemos a definir nuevas clases: $2 := 1^b$ y $3 := 2^b$, formando el siguiente triángulo cerrado de color azul:



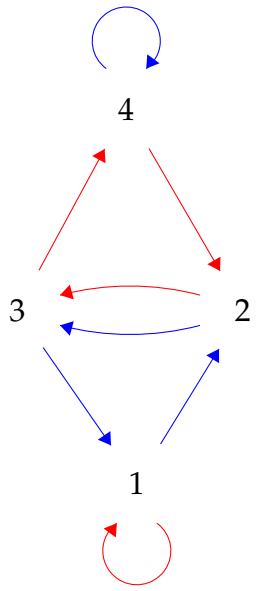
La segunda relación $b^3 = 1$ ahora sí se verifica en el vértice 1, pero no ocurre lo mismo con $(ab)^2 = 1$, obligándonos a definir $3 := 2^a$ y completar el escaneo satisfactorio del vértice 1. De nuevo, las flechas rojas entrando y saliendo de los vértices 2 y 3 nos indican que el algoritmo no ha terminado y que se deben realizar nuevas definiciones para completar el grafo.



3. Con el resto de vértices se sigue un proceso análogo al realizado con el vértice 1. Ambos vértices 2 y 3, satisfacen $b^3 = 1$; sin embargo no ocurre lo mismo con el resto de relaciones, por lo que se procede a realizar las definiciones $4 := 3^a$ y $2 := 4^a$, obteniendo así el siguiente grafo:



4. Concluimos que la definición que falta debe ser $4 := 4^b$, teniendo así un escaneo completo en todos los vértices y obteniendo un grafo completo, ya que cada vértice satisface las tres relaciones, y dada una etiqueta $x \in X$, cada vértice tiene una arista saliendo y otra entrado con ésta.



El vértice 1 hace referencia a la clase $H \in G/H$. El vértice 2 a la clase Hb, el vértice 3 a la clase $Hb^{-1} = Hb^2$, y por último, el vértice 4 a la clase $Hb^2a = Hba^2$.

Como consecuencia, se obtiene un algoritmo que resuelve el Problema de Palabras. Dos palabras w, w' dadas en el alfabeto $X^{\pm 1}$ representan el mismo elemento si partiendo del vértice 1 se sigue un recorrido en el que terminan en el mismo vértice.

En ejemplos más complicados es de gran ayuda usar una tabla de relator para cada relación que defina al grupo para así deducir nuevas definiciones y agilizar el proceso. En la cabecera de cada tabla se coloca la relación, la primera posición de cada nueva fila hace referencia a cada clase de G/H. Por otro lado, como el recorrido debe acabar

en el mismo vértice, la última posición de la fila debe coincidir con la primera. El resto de entradas de la tabla se deben rellenar si están definidas o se pueden deducir.

Para ilustrar el uso de estas tablas, repitamos el ejemplo anterior. En primer lugar, comenzamos definiendo la clase $1 := 1^a$. Para reflejar esto en las tablas de relatores, se añade la clase 1 como una nueva fila. Como el recorrido debe acabar en el mismo vértice, la última posición de cada fila también debe coincidir con la primera; en este caso, la clase 1. En el resto de entradas se opera de forma parecida a una tabla de multiplicar.

a	a	a
1	1	1

b	b	b
1		1

a	b	a	b
1	1		1

La tabla de relator para *aaa* ya está completa, pero no las otras dos, por lo que se siguen definiendo clases hasta completarlas. Definimos $2 := 1^b$ y, al hacer uso de una nueva clase, se añade una nueva fila y se completan las entradas cuyo valor conocemos:

a	a	a
1	1	1
2		2

b	b	b
1	2	1
2		2

a	b	a	b
1	1	2	1
2		1	2

De nuevo, $3 := 2^b$ y se deduce que $3^b = 1$. Como consecuencia, volvemos a añadir una nueva fila que hace referencia a la clase 3. El escaneo es ahora satisfactorio en la tabla de relator *bbb* para las 3 clases.

a	a	a
1	1	1
2		2
3		3

b	b	b
1	2	3	1
2	3	1	2
3	1	2	3

a	b	a	b
1	1	2	3	1
2		1	1	2
3			2	3

Se sigue el mismo proceso realizado para el grafo de Schreier, definiendo las clases necesarias, $3 := 2^a$, $4 := 3^a$, $2 := 4^a$ y $4 := 4^b$, y completando así todas las tablas de relatores.

a	a	a
1	1	1
2	3	4
3	4	2
4	2	3

b	b	b
1	2	3	1
2	3	1	2
3	1	2	3
4	4	4	4

a	b	a	b
1	1	2	3	1
2	3	1	1	2
3	4	4	2	3
4	2	3	4	4

Siguiendo el proceso que se realizó en el grafo de Schreier, no es dificil comprobar que las tablas de relatores anteriores son equivalentes al grafo obtenido.

Sabemos que el índice de $[G : H]$ coincide con el número de clases laterales, en total 4. Podemos afirmar que $|H| = 3$ ya que el generador a de G no es el trivial. Por otro lado, aplicando el *Teorema de Lagrange*:

$$|G| = [G : H] \cdot |H| = 4 \cdot 3 = 12.$$

Una vez ejecutado el algoritmo sobre un grupo G y un subgrupo $H \leq G$, obtenemos una tabla de clases de G sobre H, que es equivalente a la acción de G sobre las clases de G/H por la multiplicación a la derecha. Podemos obtener de forma sencilla la representación por permutaciones del grupo G.

En nuestro ejemplo, tenemos: $G := \langle a, b \mid a^3, b^3, (ab)^2 \rangle$, $H = \langle a \rangle$ y $\Omega = \{1, 2, 3, 4\}$. Definimos el homomorfismo $\varphi \colon G \to S(\Omega)$ del siguiente modo:

$$\begin{aligned} \varphi \colon G &\longrightarrow S(\Omega) \\ a &\mapsto \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 4 & 2 \end{pmatrix} = (234) \\ b &\mapsto \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix} = (123) \end{aligned}$$

Basta con observar en el grafo de Schreier el recorrido que sigue la acción para cada generador del grupo. En este ejemplo, tenemos que el grupo está generado por dos ciclos (234) y (123), que se conocen como generadores de Schreier, y claramente generan al grupo Alternado de orden 12. Por tanto:

$$G \cong \langle (234), (123) \rangle = A_4.$$

2.3.1.2 *Coincidencias*

En el proceso de definición de las diferentes clases, se puede dar la situación de que dos clases distintas resultan ser la misma. Esto es lo que se conoce como *coincidencia*, y cuando se detecta una de ellas, se ha de reemplazar el grafo Γ por un grafo cociente que refleje dicha coincidencia. Esta es, quizás, la parte más complicada del algoritmo, y para ilustrarla, se resolverá el Ejemplo 6. Consideramos el grupo:

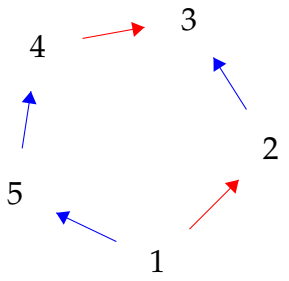
$$G = \langle X \mid R \rangle = \langle a, b \mid aba^{-1}b^{-1}b^{-1}, bab^{-1}a^{-1}a^{-1} \rangle$$

y el subgrupo trivial $H = \{1\} \leq G$. Apliquemos el *Algoritmo de Todd Coxeter*, usando el color rojo para representar la acción de a y el azul para la acción de b.

Se comienza con la clase 1 y se realizan sucesivas definiciones hasta que el primer relator se escanee por completo:

$$2 := 1^a, \quad 3 := 2^b, \quad 4 := 3^{a^{-1}}, \quad 5 := 4^{b^{-1}}.$$

A raíz de estas definiciones, se deduce que $1 = 5^{b^{-1}}$ y el grafo actual tiene forma de pentágono.



Se tendrán únicamente dos tablas de relatores, una para $aba^{-1}b^{-1}b^{-1}$ y otra para $bab^{-1}a^{-1}a^{-1}$. A partir del grafo anterior, no es difícil ver que las tablas actuales son:

a	b	a^{-1}	b^{-1}	b^{-1}	
1	2	3	4	5	1
2				3	2
3				2	3
4	3		1	5	4
5				4	5

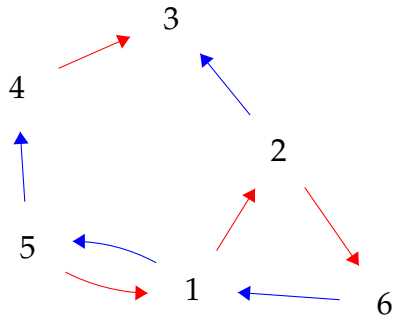
b	a	b^{-1}	a^{-1}	a^{-1}	
1	5			2	1
2	3				2
3					3
4				3	4
5	4	3	2	1	5

En este punto, el vértice 1 escanea satisfactoriamente el primer relator. Para completar la primera fila del segundo relator, deducimos $1 = 5^a$ y definimos $6 := 1^{b^{-1}}$. Pero esto último arroja también la deducción $6 = 2^a$.

a	b	a^{-1}	b^{-1}	b^{-1}	
1	2	3	4	5	1
2	6	1		3	2
3					3
4	3				4
5	1	5		4	5
6					6

b	a	b^{-1}	a^{-1}	a^{-1}	
1	5	1	6	2	1
2	3		2	1	2
3					3
4				3	4
5	4	3	2	1	5
6	1	2		2	6

El grafo de Schreier asociado al momento actual es el siguiente:



A partir de aquí es cuando nos encontraremos *coincidencias*. El proceso a seguir es el de reemplazar nuestro grafo de Schreier por un grafo cociente que refleje dicha

coincidencia. Este nuevo grafo debe satisfacer las propiedades del grafo original (que cada vértice tenga exactamente una arista de cada color entrando y otra saliendo). Para llevar esto a cabo trabajaremos con relaciones de equivalencia, donde cada una de ellas estará representada por su elemento más pequeño.

Construiremos una función $p : \{1, \ldots, 6\} \longrightarrow \{1, \ldots, 6\}$ tal que $p(x)$ es equivalente a x para todo x y $p(x) \leq x$ para todo x. La igualdad se dará si, y sólo si, x es el representante de su clase de equivalencia.

En resumen, dada una coincidencia, se deberá eliminar el vértice que tenga un mayor valor en su clase de equivalencia y transferir todas las aristas que entran y salen al vértice que tenga un menor valor en dicha clase. Por otro lado, si estamos ante una situación en la que se produzcan coincidencias consecutivas, se hará uso de una cola (*queue*) para indicar aquellas clases que se han eliminado pero aún deben ser procesadas.

Por ejemplo, la segunda fila del primer relator se escanea por completo y nos da la deducción $3 = 5^{b^{-1}}$; sin embargo, cuando añadimos una arista de color azul desde $3 \to 5$, nos damos cuenta que el 5 recibe una arista de ese mismo color del vértice 1; por ello, obtenemos la coincidencia $3 = 1$. Para empezar, definamos $p(3) := 1$ y $p(x) := x$ si $x \neq 3$. El grafo cociente tiene ahora 5 vértices: $1, 2, 4, 5$ y 6.

Consideramos las aristas que salen y entran de 3:

- `Arista azul` $2 \to 3$: se convierte en una arista azul $2 \to 1$ en el cociente. Sin embargo, el vértice 1 ya recibe una arista azul de 6, luego volvemos a obtener una coincidencia $6 = 2$. Por orden, se borra la arista $2 \to 3$, se redefine $p(6) := 2$ (para reflejar que 6 es equivalente a 2) y se añade 6 a la cola de clases.
- `Arista roja` $4 \to 3$. Al intentar añadir una arista roja de $4 \to 1$ nos encontramos con que a 1 ya le llega una arista de este color desde 5. Se trata de una coincidencia $5 = 4$. De nuevo, y por orden, se borra la arista roja $4 \to 3$, se define $p(5) := 4$ (5 es equivalente a 4), se añade 5 a la cola para procesarlo más adelante y se borra el vértice 3.

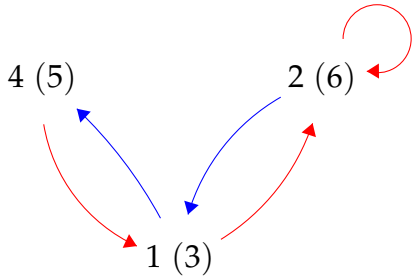
Procesamos ahora las aristas que entran y salen del vértice 6:

- `Arista azul` $6 \to 1$: como 6 es equivalente a 2, esta arista se convierte en una arista de $2 \to 1$.
- `Arista roja` $2 \to 6$: pasa a ser una arista roja que sale y entra del 2.

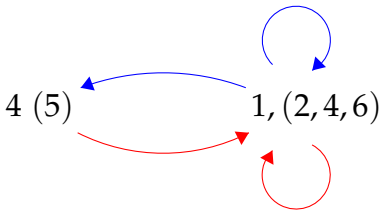
El siguiente vértice que se encontraba en la cola era 5, cuyas aristas deben procesarse:

- `Arista roja` $5 \to 1$: se convierte en una arista roja $4 \to 1$.
- `Aristas azules` $1 \to 5$ `y` $5 \to 4$: ambas pasan a ser una arista azul $1 \to 4$.

Como ya se han procesado todas las aristas de los vértices 4 y 6, se pueden eliminar, obteniendo el siguiente grafo, que contiene 3 vértices: $1, 2$ y 4:



Sin embargo, el vértice 2 tiene dos aristas rojas entrando, por lo que se obtiene la coincidencia $2 = 1$. Ahora bien, se redefine $p(2) := 1$ para indicar que el vértice 2 es equivalente a 1.



En el vértice 1 entran dos aristas de color rojo y salen otras dos de color azul, luego se redefine $p(4) := 1$ al obtener la coincidencia $4 = 1$. Siguiendo el mismo proceso para las aristas de 4, se obtiene finalmente que el grafo cociente es el vértice 1. El *Algoritmo de Todd Coxeter* ha terminado y prueba que G es el grupo trivial.

3

PRODUCTO DE GRUPOS

En esta sección se presentarán y estudiarán diferentes herramientas para la construcción de un grupo a partir de otros más simples. Dados dos grupos H y K, la forma más sencilla es considerar el producto directo $H \times K$, que es análogo al producto cartesiano de la Teoría de Conjuntos. En 3.1 recordaremos su construcción interna y externa. Más adelante, en 3.2 presentaremos y estudiaremos el producto semidirecto, sin embargo, esta construcción requiere de acciones de grupo por lo que se introducirán previamente. La documentación usada para esta sección ha sido principalmente [DF03] y [Jud17], ayudándonos en gran medida de [Pal18].

3.1 PRODUCTO DIRECTO

Definición 16. Sean $(H, \cdot_1)$ y $(K, \cdot_2)$ dos grupos. El producto directo (o simplemente producto) de ambos grupos es el producto cartesiano:

$$H \times K := \{(h,k) \mid h \in H, k \in K\},$$

dotado de la operación binaria $(\cdot)$:

$$(h_1,k_1)(h_2,k_2) := (h_1 \cdot_1 h_2, k_1 \cdot_2 k_2), \text{ para todo } h_1,h_2 \in H, k_1,k_2 \in K.$$

Denotando $G = H \times K$, como $(H, \cdot_1)$ y $(K, \cdot_2)$ son grupos entonces $(G, \cdot)$ también lo es y su orden será $|H| \cdot |K|$. La identidad es $(1_H, 1_K)$; para cada $h \in H, k \in K$, su elemento inverso vendrá dado por (h^{-1}, k^{-1}), y por último, la propiedad asociativa en G se cumple por la asociatividad de H y K.

Por otro lado, los conjuntos $\{(h,1) \mid h \in H\}$ y $\{(1,k) \mid k \in K\}$ son subgrupos normales de G y se identifican con H y K, respectivamente, es decir:

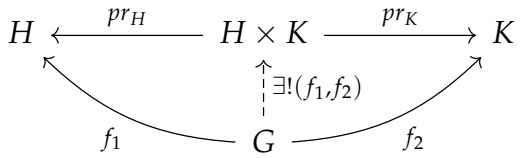
$$H \cong \{(h,1) \mid h \in H\} \quad y \quad K \cong \{(1,k) \mid k \in K\}.$$

Si H y K son grupos abelianos, está claro que su producto $H \times K$ también es un grupo abeliano.

Teorema 16 (Propiedad Universal)**.** *Sea $H \times K$ un producto de grupos. Consideramos las proyecciones:*

$$\begin{array}{ccccc} H & \xleftarrow{pr_H} & H\times K & \xrightarrow{pr_K} & K \\ h & \longleftarrow & (h,k) & \longmapsto & k \end{array}$$

Dado cualquier grupo G y dos morfismos de grupos $f_1\colon G \to H$ y $f_2\colon G \to K$, entonces existe un único morfismo (f_1, f_2) tal que $(f_1, f_2)(g) := (f_1(g), f_2(g))$, que hace al siguiente diagrama conmutativo:



Ejemplo 7. *El grupo $\mathbb{Z}_2 \times \mathbb{Z}_2$ es conocido como grupo de Klein y es el producto directo del grupo cíclico de orden dos por sí mismo:*

$$K = \{(0,0), (0,1), (1,0), (1,1)\},$$

Este grupo en notación multiplicativa puede representarse como:

$$K = \{1, a, b, ab\}.$$

donde a y b conmutan y en la que cada elemento (menos la identidad) tiene orden 2 y es inverso de sí mismo. Otra forma comúnmente conocida es mediante permutaciones, el subgrupo de A_4 conocido como Vierergruppe (en alemán) y denotado con la letra V:

$$V = \{1, (12)(3,4), (13)(24), (14)(23)\}.$$

Proposición 1. *El orden de cada elemento $(h,k) \in H \times K$ es el mínimo común múltiplo de los órdenes de h y k:*

$$|(h,k)| = mcm(|h|, |k|).$$

En particular, si $|h|$ y $|k|$ son primos relativos, entonces el orden de (h,k) es el producto de los órdenes de h y k.

Proposición 2. *Consideramos el producto de grupos cíclicos $\mathbb{Z}_m \times \mathbb{Z}_n$, se tiene:*

$$\mathbb{Z}_m \times \mathbb{Z}_n \cong \mathbb{Z}_{mn} \iff mcd(n,m) = 1\,.$$

La construcción realizada en la Definición 16 caracteriza al producto de grupos de manera externa. En cambio, en ocasiones resulta de interés estudiar si un grupo es producto directo de dos subgrupos suyos, es decir, establecer un criterio para dar un isomorfismo entre el grupo y el producto directo de sus subgrupos. La siguiente Proposición 3 caracterizará el producto directo de forma interna.

Proposición 3. *Sea G un grupo y $H, K \leq G$ subgrupos normales satisfaciendo:*

1. $H, K \trianglelefteq G$,
2. $G = HK$,
3. $H \cap K = \{1\}$.

entonces:

$$G \cong H \times K\,.$$

Para el caso general, se construye el producto directo de igual forma. En primer lugar, definiremos la caracterización externa del producto directo de grupos, y en la Proposición 4 destacaremos propiedades del producto que no serán demostradas por ser estándar.

Sean H_i , $i \in I = \{1, 2, \ldots, n\}$ grupos. Definimos el producto directo de esta familia de grupos como el producto cartesiano:

$$H_1 \times H_2 \times \cdots \times H_n := \prod_{i \in I} H_i = \{(h_i)_{i \in I} \mid h_i \in H_i\},$$

dotado de la operación

$$(h_1, h_2, \cdots, h_n)(h_1', h_2', \cdots, h_n') = (h_i)_{i \in I}(h_i')_{i \in I} := (h_i h_i')_{i \in I}$$

Proposición 4.

1. *Sean $H_1, \ldots, H_n$ subgrupos de un grupo G. El grupo G es producto directo de sus subgrupos $H_1, \ldots, H_n$ si:*
   a) $H_i \trianglelefteq G,\ \forall i = 1, \ldots, n.$
   b) $H_i \cap (\bullet_{j \neq i} H_j) = \{1\},\ \forall i = 1, \ldots, n.$
   c) $H_1 \cdots H_n = G.$
2. *Si $H_1, H_2, \ldots, H_n$ son grupos finitos, su producto directo es un grupo de orden:*

$$|H_1| \cdot |H_2| \cdots |H_n|\,.$$

3.2 PRODUCTO SEMIDIRECTO

En esta sección presentaremos y desarrollaremos el producto semidirecto para la construcción de un grupo G a partir de dos grupos H y K. Su construcción requiere de acciones de grupo, que se introdujeron en la Sección 1.3.1. Este concepto es clave para dos de los teoremas más importantes del álgebra abstracta, el *Teorema de Cayley* 1 y los *Teoremas de Sylow* 7.

Como hemos comentado anteriormente, dar una acción de grupo (9) es equivalente a dar un homomorfismo de grupos $G \to S(X)$, el grupo de Permutaciones del conjunto X. Ahora bien, en vez de considerar un conjunto X, podemos tomar un grupo H y restringirnos a las acciones de $G \to \mathrm{Aut}(H)$ que son compatibles con una estructura de grupo, es decir, acciones que satisfagan las propiedades 1 y 2.

El producto semidirecto de dos grupos H y K es una generalización del producto directo en la que la condición de normalidad de ambos grupos H y K se encuentra "relajada". Esta herramienta nos permitirá construir un grupo más grande G, de tal manera que contenga subgrupos isomorfos a H y K, al igual que en el producto directo. En este caso, H será normal en G, pero el subgrupo K no tiene por qué serlo. Así, por ejemplo, podremos construir grupos no abelianos incluso si H y K son abelianos.

Teorema 17. *Sean K un grupo y H un K-grupo, llamemos $\varphi\colon K \to \mathrm{Aut}(H)$ al morfismo inducido por la acción:*

$$\begin{aligned} \varphi\colon K &\longrightarrow \mathrm{Aut}(H) \\ k &\longmapsto \varphi(k) = \varphi_k\colon \quad H \longrightarrow H \\ &\qquad\qquad\quad h \longmapsto {}^k h = khk^{-1} \end{aligned}$$

y definamos G como el conjunto de los pares (h,k) con $h \in H$, $k \in K$ dotado con la operación:

$$(h_1,k_1)(h_2,k_2) = (h_1 \cdot \varphi_{k_1}(h_2), k_1k_2), \quad \forall\, h_1,h_2 \in H,\ \forall\, k_1,k_2 \in K.$$

Entonces, se cumple:

1. *G es un grupo de orden $|H| \cdot |K|$.*
2. *Los conjuntos $\{(h,1) \mid h \in H\}$ y $\{(1,k) \mid k \in K\}$ son subgrupos de G y las aplicaciones $h \mapsto (h,1)$, $h \in H$ y $k \mapsto (1,k)$, $k \in K$ son isomorfismos de esos dos grupos en H y en K, es decir:*

$$H \cong \{(h,1) \mid h \in H\} \quad y \quad K \cong \{(1,k) \mid k \in K\}.$$

Identificando H y K con sus isomorfismos en G descritos en 2. y la acción por conjugación en G, se tiene:

3. $H \trianglelefteq G$,

4. $H \cap K = 1$,
5. $G = KH$.

Demostración.

No es complicado probar que G es un grupo:

- El elemento neutro de G es $(1_K, 1_H)$ puesto que:

$$(k,h)(1_K,1_H) = (k \cdot \varphi_h(1_K), h \cdot 1_H) = (k,h).$$
$$(1_K,1_H)(k,h) = (1_K \cdot \varphi_{1_H}(k), 1_H \cdot h) = (k,h).$$

- Los elementos inversos vienen dados por $(k,h)^{-1} = (\varphi_{h^{-1}}(k^{-1}), h^{-1})$:

$$\begin{aligned}(k,h)\cdot(k,h)^{-1} &= (k,h)\cdot(\varphi_{h^{-1}}(k^{-1}),h^{-1}) = (k\cdot\varphi_h(\varphi_{h^{-1}}(k^{-1})),hh^{-1}) = \\ &= (k\cdot\varphi_{hh^{-1}}(k^{-1}),hh^{-1}) = (kk^{-1},hh^{-1}) = (1_K,1_H).\\ (k,h)^{-1}\cdot(k,h) &= (\varphi_{h^{-1}}(k^{-1}),h^{-1})\cdot(k,h)(\varphi_{h^{-1}}(k^{-1})\varphi_{h^{-1}}(k),hh^{-1}) = \\ &= (\varphi_{h^{-1}}(k^{-1}k),1_H) = (\varphi_{h^{-1}}(1_K),1_H) = (1_K,1_H).\end{aligned}$$

- Asociatividad:

$$\begin{aligned}\big[(k_1,h_1)\cdot(k_2,h_2)\big]\cdot(k_3,h_3) &= (k_1\cdot\varphi_{h_1}(k_2),h_1\cdot h_2)\cdot(k_3,h_3) = \\ =(k_1\cdot\varphi_{h_1}(k_2)\cdot\varphi_{h_1h_2}(k_3),h_1\cdot h_2\cdot h_3) &= (k_1\cdot\varphi_{h_1}(k_2)\varphi_{h_1h_2}(k_3),h_1\cdot h_2\cdot h_3) = \\ =(k_1\cdot\varphi_{h_1}(k_2\cdot\varphi_{h_2}(k_3)),h_1\cdot h_2\cdot h_3) &= (k_1,h_1)\cdot(k_2\varphi_{h_2}(k_3),h_2h_3) = \\ &= (k_1,h_1)\cdot\big[(k_2,h_2)\cdot(k_3,h_3)\big].\end{aligned}$$

El orden del grupo G claramente es el producto del orden de H y K, que prueba 1.

Sean $\widetilde{H} = \{(h,1) \mid h \in H\}$ y $\widetilde{K} = \{(1,k) \mid k \in K\}$, se tiene:

$$(x,1)(y,1) = (xy,1), \quad \forall x,y \in H.$$

Además,

$$(1,x)(1,y) = (1,xy), \quad \forall x,y \in K.$$

que muestran que $\widetilde{H}$ y $\widetilde{K}$ son subgrupos de G cumpliendo el punto 5., que las aplicaciones de 2. son isomorfismos y que $\widetilde{H} \cap \widetilde{K} = 1$ (4.).

Por último, bajo los isomorfismos de 2., $K \leq N_G(H)$. Como $G = NK$ y $H \leq N_G(H)$, se tiene que $N_G(H) = G$, o en otras palabras, $H \trianglelefteq G$, que demuestra el punto 3. y completa la prueba. □

Definición 17. Sean H y K grupos y $\varphi\colon K \to \mathrm{Aut}(H)$ un homomorfismo. El grupo G descrito en el Teorema 17 se denomina *producto semidirecto* de H y K con respecto a φ y se denota por $H \rtimes_\varphi K$.

Comentario. Cuando la acción está clara el producto semidirecto se denotará $H \rtimes K$. Esta notación se ha elegido para recordarnos que H es normal en $H \rtimes K$ y que la construcción del producto semidirecto no es simétrica en H y K (a diferencia del producto directo).

Al igual que en el producto directo, el producto semidirecto admite una caracterización de forma interna, como veremos en el Teorema 18, que se enunciará y demostrará a continuación:

Teorema 18. *Sea G un grupo con subgrupos H y K satisfaciendo:*

1. $H \trianglelefteq G$,
2. $G = HK$,
3. $H \cap K = 1$.

y sea $\varphi\colon K \to \mathrm{Aut}(H)$ el homomorfismo definido por la aplicación que envía $k \in K$ al automorfismo de conjugación de k en H, es decir, $\varphi_k(h) = khk^{-1}$, $\forall h \in H$. Entonces:

$$G \cong H \rtimes_\varphi K\,.$$

Demostración.

Definimos la aplicación:

$$\begin{aligned} f\colon H \rtimes_\varphi K &\to G \\ (h,k) &\mapsto hk \end{aligned}$$

- La sobreyectividad de f es evidente ya que $G = HK$.
- Veamos que f es inyectiva. Sean $h_1, h_2 \in H$, $k_1, k_2 \in K$. Si $f(h_1,k_1) = f(h_2,k_2)$, entonces $h_1k_1 = h_2k_2$. Se tiene que $h_2^{-1}h_1 = k_2k_1^{-1} \in H \cap K = 1$, luego $h_1 = h_2$ y $k_1 = k_2$.
- f es un homomorfismo:

$$\begin{aligned} f((h_1,k_2),(h_2,k_2)) = f(h_1 \cdot \varphi_{k_1}(h_2), k_1k_2) = h_1(k_1h_2k_1^{-1})k_1k_2 = \\ = h_1k_1h_2k_2 = f(h_1,k_1)f(h_2,k_2). \end{aligned}$$

□

Comentario. En el Teorema 17 se ha construido el producto semidirecto externo de dos grupos a partir de un grupo K, un K-grupo H y la acción de conjugación de K en $\mathrm{Aut}(H)$. En cambio, a veces es de utilidad estudiar si un grupo G satisface las hipótesis del Teorema 18 para ser producto semidirecto de dos subgrupos. A continuación, veremos algunos ejemplos:

Ejemplo 8. *Veamos que S_n es producto semidirecto interno de A_n y K, $S_n \cong A_n \rtimes K$, donde $K = \{1, (12)\}$.*

1. *El índice $[S_n : A_n] = 2$, por lo que $A_n \trianglelefteq S_n$ y $A_nK \leq S_n$.*
2. *Trivialmente, $A_n \cap K = \{1\}$. De hecho, se cumple:*

$$|A_nK| = \frac{|A_n| \cdot |K|}{|A_n \cap K|} = \frac{\frac{n!}{2} \cdot 2}{1} = n!$$

y se tiene que $|A_nK| = |S_n|$. Luego A_n y K satisfacen las condiciones del Teorema 18 y podemos afirmar que:

$$S_n \cong A_n \rtimes K\,.$$

Ejemplo 9. *El grupo Diédrico $D_n =< \varphi\,, \sigma >$ es producto semidirecto interno. Se cumple:*

1. *El subgrupo formado por el conjunto de las rotaciones $< \varphi >$ es un subgrupo normal de D_n. Por otro lado, $< \sigma > \leq D_n$.*
2. *$D_n =< \varphi > \cdot < \sigma >$.*
3. *$< \varphi > \cap < \sigma >= \{1\}$.*

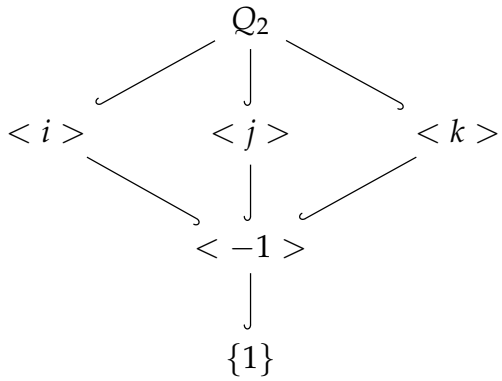
Ambos subgrupos cumplen las condiciones del Teorema 18 y, por tanto:

$$D_n =< \varphi > \rtimes < \sigma > \ .$$

Definición 18. Sea H un subgrupo de un grupo G. Un subgrupo K de G es *complemento* de H en G si $G = HK$ y $H \cap K = 1$.

Con la Definición 18, el criterio para reconocer un producto semidirecto se reduce a la existencia de un subgrupo que sea complemento para algún subgrupo normal propio de G.

Ejemplo 10. *No siempre un grupo puede expresarse como producto semidirecto de dos de sus subgrupos, como es el caso del grupo de los Cuaternios Q_2. Veamos el retículo de subgrupos:*



donde:

$$< i > = \{1, -1, i, -i\}, \qquad < j >= \{1, -1, j, -j\},$$
$$< k > = \{1, -1, k, -k\}, \qquad < -1 >= \{1, -1\}.$$

Dado cualquier grupo $H \leq Q_2$ (que además es normal), no existe otro subgrupo complemento (18) de H en Q_2 que satisfaga las condiciones del Teorema 18.

Proposición 5. *Sean H y K grupos y $\varphi\colon K \to \mathrm{Aut}(H)$ un homomorfismo. Las siguientes enunciados son equivalentes:*

(1) La aplicación identidad entre $H \rtimes K$ y $H \times K$ es un homomorfismo de grupos.

(2) φ es el homomorfismo trivial de K en $\mathrm{Aut}(H)$.

(3) $K \trianglelefteq H \rtimes K$.

Demostración.

(1) $\Rightarrow$ (2) Por la operación definida en $H \rtimes K$:

$$(h_1, k_1)(h_2, k_2) = (h_1 \cdot \varphi_{k_1}(h_2), k_1 k_2) \overset{(1)}{=} (h_1 h_2, k_1 k_2), \quad \forall h_1, h_2 \in H,\ \forall k_1, k_2 \in K.$$

se tiene que $\varphi_{k_1}(h_2) = h_2,\ \forall h_2 \in H, \forall k_1 \in K$ y K actúa trivialmente sobre H.

(2) $\Rightarrow$ (3) Si φ es el homomorfismo trivial, entonces la acción de K sobre H es trivial por lo que por el Teorema 17 los elementos de H conmutan con los de K. En particular, $H \leq N_G(K)$ y $G = HK \leq N_G(K)$.

(3) $\Rightarrow$ (1) Si K es normal en $H \rtimes K$ entonces $\forall h \in H$ y $k \in K$, $[h, k] \in H \cap K = 1$. Así, $hk = kh$ y la acción de K en H es trivial. La multiplicación en el producto semidirecto coincide con la del producto directo:

$$(h_1, k_1)(h_2, k_2) = (h_1 h_2, k_1 k_2), \quad \forall h_1, h_2 \in H,\ \forall k_1, k_2 \in K.$$

□

Consideramos H y K dos grupos, $\varphi\colon K \to \mathrm{Aut}(H)$ un homomorfismo de K en los automorfismos de H y $H \rtimes K$ su producto semidirecto. Por el Teorema 17, $H \trianglelefteq H \rtimes K$, pero no necesariamente K. Si $H \rtimes K$ es abeliano, entonces cada subgrupo es normal por lo que si K es normal, entonces por la Proposición 5, φ sería el homomorfismo trivial y el producto semidirecto coincidiría con el producto directo, es decir:

$$H \times K \cong H \rtimes K.$$

Ejemplo 11. *Consideramos los grupos cíclicos $C_2 = \langle a \rangle$ y $C_3 = \langle b \rangle$. Estudiemos los posibles productos semidirectos:*

- *$C_2 \rtimes C_3$, el único homomorfismo $\varphi\colon C_3 \to \mathrm{Aut}(C_2)$ es el trivial por lo que el producto semidirecto coincide con el producto directo.*
- *$C_3 \rtimes C_2$, se tiene que $C_3 \trianglelefteq C_3 \rtimes C_2$ y, por tanto:*

$$\begin{aligned} \varphi\colon C_2 &\longrightarrow \mathrm{Aut}(C_3) \\ & \qquad C_3 \longrightarrow C_3 \\ a \longmapsto & \quad b \longmapsto \begin{cases} b \\ b^2 \end{cases} \end{aligned}$$

Hay dos acciones posibles, por lo que existirán dos productos semidirectos:

$$\varphi_a(b) = {}^a b = b \Longrightarrow C_3 \rtimes C_2 \cong C_3 \times C_2. \tag{6}$$

$$\varphi_a(b) = {}^a b = b^2 \Longrightarrow C_3 \rtimes C_2 \cong D_3. \tag{7}$$

En (6), la acción es trivial por lo que por la Proposición 5 el producto semidirecto coincide con el producto directo. Por la Proposición 2, el grupo es cíclico de orden 6.
En (7), el grupo obtenido no es abeliano por lo que debe ser isomorfo a D_3. *No es difícil aplicar el Teorema de Dyck 15 para dar dicho isomorfismo.*

4

TÉCNICAS DE CLASIFICACIÓN DE GRUPOS

El producto semidirecto nos ofrece una una herramienta potente para la clasificación de grupos, sin embargo, no funciona para todo grupo de orden n. Por ejemplo, en el Ejemplo 10 hemos visto que el grupo Q_2 no puede expresarse como producto semidirecto (interno) de dos de sus subgrupos. De hecho, es una herramienta que no funciona para grupos que son potencia alta de un número primo. La documentación usada para este desarrollo ha sido principalmente [JM17] y [Pal18], y en menor medida, de [DF03].

Aplicaremos el Teorema 18 para clasificar grupos de orden n para algunos valores específicos de n. La idea a seguir es el siguiente procedimiento:

1. Mostrar que cada grupo de orden n tiene subgrupos propios H y K que satisfacen las hipótesis del Teorema 18.
2. Encontrar todos los isomorfismos posibles para H y K.
3. Para cada pareja H, K del paso anterior, encontrar todos los posibles homomorfismos $\varphi\colon K \to \mathrm{Aut}(H)$.
4. Para cada terna H, K, φ del paso anterior, contruir el producto semidirecto $H \rtimes K$ y determinar cuáles de ellos son isomorfos, obteniendo así una lista de grupos e isomorfismos de orden n.

A modo de ejemplo y siguiendo el procedimiento descrito anteriormente, en el Ejemplo 12 se determinarán todos los grupos de orden 12 salvo isomorfismos. Pero antes, enunciamos bajo la siguiente Proposición 6 algunas propiedades que necesitaremos.

Proposición 6.

1. *Sea p un primo con $p \neq 2$ y sea $n \in \mathbb{Z}$. Entonces $\mathrm{Aut}(\mathbb{Z}_p) \cong \mathbb{Z}_{p-1}$. En el caso general, se cumple $\mathrm{Aut}(\mathbb{Z}_{p^n}) \cong \mathbb{Z}_{p^{n-1}(p-1)}$.*
2. *Sea p un primo y sea V un grupo abeliano tal que para todo $v \in V$, $pv = 0$. Si $|V| = p^n$, entonces V es un espacio vectorial de dimensión n sobre el cuerpo $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$. Además, $\mathrm{Aut}(V) \cong GL(V) \cong GL_n(\mathbb{F}_p)$, donde $|\mathrm{Aut}(V)| = (p^n - 1)(p^n - p)\cdots(p^n - p^{n-1})$.*

3. *Sean* $\varphi\colon C_n\langle x\rangle \longrightarrow \mathrm{Aut}(H)$, $i = 1,2$ *acciones de grupos. Si* $\varphi_1(x)$ *y* $\varphi_2(x)$ *son conjugados, entonces:*

$$C_n \rtimes_{\varphi_1} H \cong C_n \rtimes_{\varphi_2} H\,.$$

Ejemplo 12. *Sea* G *con* $|G| = 12 = 2^2 \cdot 3$. *Denotamos por* n_2 *el* n^o *de 2-subgrupos de Sylow y por* n_3 *el número de 3-subgrupos de Sylow, entonces:*

$$\left.\begin{array}{l} n_2 \equiv 1\, mod(2) \\ n_2 \,|\, 3 \end{array}\right\} \Rightarrow n_2 = 1,3 \quad y \qquad \left.\begin{array}{l} n_3 \equiv 1\, mod(3) \\ n_3 \,|\, 4 \end{array}\right\} \Rightarrow n_3 = 1,4\,.$$

Consideramos H *un 2-subgrupo de Sylow y* K *un 3-subgrupo de Sylow. El caso* $n_2 = 3$ *y* $n_3 = 4$ *no se puede dar ya que existirían* $4 \times (3-1) = 8$ *elementos de orden 3 y más de 3 elementos de orden 2 o 4, lo que sería una contradicción ya que* $|G| = 12$.

Como en el resto de casos alguno de los p-subgrupos es normal, se tiene que $HK = G$. *Además,* $H \cap K = \{1\}$ *por lo que se cumplen las condiciones del Teorema 18 y* G *se puede expresar como producto semidirecto. Distinguimos los siguientes casos:*

- $n_2 = 1$ *y* $n_3 = 1$. *Ambos subgrupos son normales luego por la Proposición 5,* G *es producto directo de* H *y* K*:*

$$G = H \times K \cong \mathbb{Z}_4 \times \mathbb{Z}_3 \cong \mathbb{Z}_{12}\,.$$

- $n_2 = 1$ *y* $n_3 = 4$. *Se tiene que* H *es un subgrupo normal y* $G = H \rtimes_\varphi K$. *Estudiamos los posibles homomorfismos* $\varphi\colon K \to \mathrm{Aut}(H)$. *Como* $|H| = 4$, *tenemos que distinguir dos casos:*

  - *Si* $H \cong \mathbb{Z}_4$, *entonces por la Proposición 1,* $\mathrm{Aut}(H) \cong \mathbb{Z}_2$ *y el único homomorfismo* $\varphi\colon K \to \mathrm{Aut}(H)$ *es el trivial, por tanto:*

$$G \cong H \rtimes_\varphi K \cong H \times K \cong \mathbb{Z}_{12}\,.$$

  - *Si* $H \cong \mathbb{Z}_2 \times \mathbb{Z}_2$, *entonces* $\mathrm{Aut}(H) \cong GL_2(\mathbb{Z}_2) \cong S_3$ *y existen tres posibles morfismos* $\varphi\colon K \to S_3$. *Uno de ellos debe ser el trivial, mientras que los otros dos, por la Proposición 3, dan lugar a productos semidirectos que son isomorfos ya que tienen imágenes conjugadas. Escribimos* $K = \langle x\rangle$ *y* $H = \langle y\rangle \times \langle z\rangle$.

$$\varphi_0\colon K \to \mathrm{Aut}(H);\ \varphi_0(x) = Id$$
$$\varphi_1\colon K \to \mathrm{Aut}(H);\ \varphi_1(x)(y) = yz,\ \varphi_1(x)(z) = y$$

  El morfismo φ_0 *da lugar al producto directo de grupos:*

$$G \cong K \rtimes_{\varphi_0} H \cong K \times H \cong \mathbb{Z}_{12}\,.$$

  Mientras que φ_1 *resulta en el grupo Alternado* A_4*:*

$$G \cong H \rtimes_{\varphi_1} K \cong \langle x,y,z \mid x^3 = y^2 = z^2 = (yz)^2 = 1,\ xyx^{-1} = yz,\ xzx^{-1} = y\rangle \cong A_4.$$

- *$n_2 = 3$ y $n_3 = 1$. Se tiene que K es un subgrupo normal de G, por lo que G es producto semidirecto $K \rtimes_\varphi H$ con $\varphi\colon H \to \mathrm{Aut}(K)$. De igual modo, como $|H| = 4$, distinguimos dos casos:*

  - *Si $H \cong \mathbb{Z}_4$. Se cumple que $\mathrm{Aut}(K) \cong \mathbb{Z}_2$ por la Proposición 1, por lo que existen dos homomorfismos. Escribimos $H = \langle y \rangle$ y $K = \langle x \rangle$, entonces:*

$$\varphi_0\colon H \to \mathrm{Aut}(K);\ \varphi_0(y) = Id$$
$$\varphi_1\colon H \to \mathrm{Aut}(K);\ \varphi_1(y)(x) = x^2 = x^{-1}$$

    El primer morfismo da lugar al producto semidirecto $K \rtimes_{\varphi_0} H$, que por la Proposición 5, coincide con el producto directo de grupos:

$$G \cong \langle x, y \mid x^3 = x^4 = 1, yxy^{-1} = x \rangle \cong \mathbb{Z}_3 \times \mathbb{Z}_4 \cong \mathbb{Z}_{12}.$$

    Mientras que el segundo morfismo nos da el producto semidirecto $K \rtimes_{\varphi_1} H$:

$$G \cong \langle x, y \mid x^3 = y^4 = 1, yxy^{-1} = x^{-1} \rangle \cong Q_3\,.$$

  - *Si $H \cong \mathbb{Z}_2 \times \mathbb{Z}_2$. De igual modo, $\mathrm{Aut}(K) \cong \mathbb{Z}_2 = \langle x \rangle$ y escribimos $H = \langle y \rangle \times \langle z \rangle$. En este caso existirán dos morfismos distintos:*

$$\varphi_0\colon H \to \mathrm{Aut}(K);\ \varphi_0(y)(x) = \varphi_0(z)(x) = Id$$
$$\varphi_1\colon H \to \mathrm{Aut}(K);\ \varphi_1(y)(x) = x^{-1}, \varphi_1(z)(x) = Id$$

    El primer morfismo de grupos resulta en el producto semidirecto $K \rtimes_{\varphi_0} H$, que es isomorfo a:

$$G \cong \langle x, y, z \mid x^3 = y^2 = z^2 = 1, yzy^{-1} = z, yxy^{-1} = x, zxz^{-1} = x \rangle \cong \mathbb{Z}_6 \times \mathbb{Z}_2.$$

    Mientras que el segundo producto semidirecto $K \rtimes_{\varphi_1} H$ nos da el grupo:

$$G \cong \langle x, y, z \mid x^3 = y^2 = z^2 = 1, yzy^{-1} = z, yxy^{-1} = x^{-1}, zxz^{-1} = x \rangle \cong D_6.$$

Como conclusión, tenemos que los grupos no abelianos de orden 12 salvo isomorfismos son: A_4, Q_3 y D_6, mientras que los grupos abelianos son: $\mathbb{Z}_{12}$ y $\mathbb{Z}_6 \times \mathbb{Z}_2$.

Comentario. El proceso de clasificación de grupos para todo n es costoso y llevaría mucho tiempo clasificar todos los grupos uno a uno. Además, se ha de tener en cuenta que existen grupos que no se pueden clasificar usando esta herramienta. Por estas razones, muchos matemáticos empezaron a estudiar patrones que siguen diferentes grupos, como por ejemplo Hölder, quien realizó diferentes estudios para clasificar grupos cuyo orden es producto de números primos.

Por consiguiente, en las siguientes secciones nos centraremos en determinar grupos que sigan patrones parecidos. Consideramos p, q primos: Nos basaremos en [BL18] para clasificar en 4.1 los grupos de orden p y p^2. Continuaremos en 4.2 clasificando grupos de orden pq con $p < q$, donde centraremos nuestra atención en el caso particular de grupos de orden $2p$. Por último, nos basaremos en [DF03] para determinar en el Teorema 21 todos los grupos de orden p^3.

4.1 GRUPOS DE ORDEN p Y p^2

Para clasificar grupos de orden p y p^2, enunciaremos los Teoremas 19 y 20, respectivamente. Se tratan de teoremas básicos que se dan como normal general en cualquier curso sobre Teoría de Grupos. Sin embargo, los demostraremos basándonos en [BL18] ya que nos serán de mucha utilidad en las siguientes secciones para clasificar grupos cuya construcción es más compleja.

Teorema 19. *Sea G un grupo de orden p primo, entonces G es cíclico: $G \cong C_p$.*

Demostración. Consideramos $g \in G$ distinto a la identidad. Por el *Teorema de Lagrange* 3, $|g| \mid p$, pero como $g \neq 1_G$, se tiene que $|g| = p$, y por tanto, $G = \langle g \rangle$, luego:

$$G \cong C_p \, .$$

□

Teorema 20. *Sea G un grupo de orden p^2 con p primo. Entonces G es abeliano, es decir:*

$$G \cong C_{p^2} \quad o \quad G \cong C_p \times C_p \, .$$

Demostración. Por el *Teorema de Lagrange* 3, los elementos salvo la identidad deben tener orden p o p^2. Si uno de ellos tiene orden p^2 entonces G es cíclico e isomorfo a C_{p^2}. Por el contrario, si todos ellos tienen orden p, por el Primer Teorema de Sylow 1 existe $K \trianglelefteq G$ con $|K| = p$. Tomamos $h \in G \backslash K$, que tendrá orden p y consideramos $H = \langle h \rangle$. Entonces, $H \cap N = \{1_G\}$ y $HN = G$, de modo que se cumplen las condiciones del Teorema 18 y existirá un único homomorfismo $\varphi \colon H \to \mathrm{Aut}(K)$ tal que:

$$G \cong N \rtimes_\varphi H.$$

Por otro, $\mathrm{Aut}(N) \cong C_{p-1}$ por la Proposición 1 y como $(p-1)$ no divide a $p = |h|$, entonces φ debe ser trivial, y por la Proposición 5:

$$G \cong N \rtimes_\varphi H \cong N \times H \cong C_p \times C_p \, .$$

□

4.2 GRUPOS DE ORDEN pq

Sea G un grupo de orden $p \cdot q$, con $p < q$ primos. Aplicando los *Teoremas de Sylow* 7, se tiene:

$$\left.\begin{array}{l} n_p \equiv 1 \; mod\,(p) \\ n_p \mid q \end{array}\right\} \Rightarrow n_p = 1, q \qquad y \qquad \left.\begin{array}{l} n_q \equiv 1 \; mod\,(q) \\ n_q \mid p \end{array}\right\} \Rightarrow n_q = 1\,.$$

Sean P y Q un p-subgrupo y q-subgrupo de Sylow de G. Como $n_q = 1$, se tiene que $Q \trianglelefteq G$. Además, $P \cap Q = 1$ y $G = QP$ por lo que se cumplen las condiciones del Teorema 18 y G es producto semidirecto interno de Q y P.

Distinguimos los siguientes casos según el valor que tome n_p, el número de p-subgrupos de Sylow de G:

1. $n_p = 1$. En este caso, P también sería un subgrupo normal de G y, por la Proposición 5, el producto semidirecto $Q \rtimes P$ coincide con el producto directo:

$$G = Q \times P \cong C_q \times C_p \overset{(2)}{\cong} C_{pq}\,.$$

2. $n_p = q$. Se debe cumplir que $q \equiv 1 \; mod\,(p)$, o equivalentemente, $p \big| (q-1)$. Por la Proposición 1, $\mathrm{Aut}(Q) \cong \mathbb{Z}_{q-1}$, cíclico, y por el *Teorema de Cauchy* 6, $\mathrm{Aut}(Q)$ contiene un único subgrupo de orden p, lo denotamos $\langle \gamma \rangle$.
   Sea $P = \langle y \rangle$, entonces cualquier homomorfismo $\varphi \colon P \to \mathrm{Aut}(Q)$ debe aplicar el generador $y \in P$ a una potencia de γ. En total, hay p homomorfismos que vienen dados por:

$$\begin{aligned} \varphi_i \colon P &\to \mathrm{Aut}(Q) \\ y &\mapsto \gamma^i\,, \quad 0 \leq i \leq p-1\,. \end{aligned}$$

   El homomorfismo trivial φ_0, por la Proposición 5, da lugar al producto directo:

$$Q \rtimes_{\varphi_0} P \cong Q \times P\,.$$

   El resto de los $p-1$ homomorfismos de grupos dan lugar a grupos no abelianos de orden pq, que serán todos isomorfos entre sí ya que para cada φ_i existe un y_i generador de P tal que $\varphi_i(y_i) = \gamma$. Luego si $p|(q-1)$, entonces:

$$G \cong C_q \rtimes C_p \cong \langle x, y \mid x^q, y^p, yxy^{-1} = x^{-1} \rangle\,. \tag{8}$$

De esta forma, todos los grupos que tengan un orden producto de dos primos estarían clasificados. Algunos grupos serían aquellos de orden $6, 10, 14, 15, 21 \ldots$ etc.

Comentario. Si G es un grupo de orden $2p$ entonces debe ser isomorfo al grupo cíclico $\mathbb{Z}_{2p}$ o al grupo Diédrico D_p, que tendrá una presentación como (8).

Para dar este isomorfismo basta considerar una presentación del grupo Diédrico:

$$D_p = \langle \varphi, \sigma \mid \varphi^p, \sigma^2, \sigma\varphi\sigma^{-1} = \varphi^{-1} \rangle$$

y aplicar el Teorema de Dyck 15.

$$\begin{aligned} D_p &\to C_p \rtimes C_2 \\ \varphi &\mapsto x \\ \sigma &\mapsto y \end{aligned}$$

Es evidente que x e y satisfacen las relaciones de D_p. Además, ambos grupos tienen $2p$ elementos luego son isomorfos.

4.3 GRUPOS DE ORDEN p^3

En esta sección nos centraremos en los grupos cuyo orden es potencia cúbica de un número primo. Nos basaremos en las notas de [DF03] para demostrar el Teorema 21, pero antes, introduciremos la siguiente proposición:

Proposición 7. *Si G es un grupo no abeliano de orden p^3, con $p \neq 2$, entonces G es producto semidirecto de H y K, donde H es un subgrupo normal de orden p^2 y K es un subgrupo de orden p.*

Demostración. G es producto semidirecto interno de H y K ya que satisfacen las condiciones del Teorema 18. □

Teorema 21. *Sea p un primo con $p \neq 2$, entonces, salvo isomorfismos, existen 5 grupos de orden p^3.*

Demostración. Sea G un grupo con $|G| = p^3$. Si G es abeliano, entonces se tendrá:

$$G \cong \mathbb{Z}_{p^3}, \quad G \cong \mathbb{Z}_{p^2} \times \mathbb{Z}_p \quad o \quad G \cong \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p \,.$$

Si G es no abeliano, por la Proposición anterior 7, se tiene que:

$$G \cong H \rtimes K \,,$$

donde $|H| = p^2$ y $|K| = p$. Por el Teorema 20, todo grupo de orden p^2 es abeliano, luego se tendrá que:

$$H \cong \mathbb{Z}_{p^2} \quad o \quad H \cong \mathbb{Z}_p \times \mathbb{Z}_p \,.$$

1. $H \cong \mathbb{Z}_p \times \mathbb{Z}_p$ y $K \cong \mathbb{Z}_p$.

   Sea $\varphi \colon K \to H$ un homomorfismo de grupos. Por la Proposición 2, se tiene que $\mathrm{Aut}(H) \cong GL_2(\mathrm{F}_p)$, que tiene orden $(p^2-1)(p^2-p) = p(p^2-1)(p-1)$.

Como $p \mid\mid \text{Aut}(H)|$, por el *Teorema de Cauchy* 6, $\text{Aut}(H)$ tiene un único automorfismo de orden p. De este modo, hay un homomorfismo de grupos $\varphi\colon K \to \text{Aut}(H)$ no trivial donde $H \rtimes K$ es un grupo no abeliano de orden p^3.

Escribimos $K = \langle x \rangle$ y $H = \langle y \rangle \times \langle z \rangle$ y $\varphi\colon K \to \text{Aut}(H)$ vendrá dada por:

$$\varphi(x)(y) = yz,\ \varphi(x)(z) = z$$

El producto semidirecto $H \rtimes_\varphi K$ será isomorfo a:

$$G_1 = \langle x, y, z \mid x^p = y^p = z^p = 1,\ yzy^{-1} = z,\ xyx^{-1} = yz,\ xzx^{-1} = z \rangle.$$

2. $H \cong \mathbb{Z}_{p^2}$ y $K \cong \mathbb{Z}_p$.

   Sea $\varphi\colon K \to \text{Aut}(H)$ un homomorfismo de grupos. Por la Proposición 1, se tiene que $\text{Aut}(H) \cong \mathbb{Z}_{p(p-1)}$, cíclico, luego H tiene un único automorfismo de orden p. Así, hay un único homomorfismo de grupos $\varphi\colon K \to \text{Aut}(H)$ no trivial ,y por tanto, $H \rtimes K$ será un grupo no abeliano de orden p^3.

   Si $K = \langle x \rangle$ y $H = \langle y \rangle$, entonces el morfismo $\varphi\colon K \to \text{Aut}(H)$ viene dado por:

$$\varphi(x)(y) = y^{1+p}$$

   El grupo $H \rtimes_\varphi K$ tiene presentación:

$$G_2 = \langle x, y \mid y^{p^2}, x^p, xyx^{-1} = y^{1+p} \rangle.$$

Para terminar, G_1 y G_2 no son isomorfos. G_2 contiene un elemento de orden p^2 mientras que en G_1 todo elemento distinto a la identidad tiene orden p. □

Comentario. Para $p = 2$, se tiene que el grupo de los Cuaternios Q_2 tiene orden $p^3 = 8$. Sin embargo, se ha visto en el Ejemplo 10 que este grupo no es producto semidirecto interno de dos de sus subgrupos por lo que este teorema no es válido para clasificar grupos de orden 8.

Como consecuencia del Teorema 21, los grupos de orden 27, 125, 343 . . . etc estarían clasificados por ser potencia cúbica de un número primo. En resumen, se tiene:

$\lvert G \rvert$	G_1	G_2
27	$(\mathbb{Z}_3 \times \mathbb{Z}_3) \rtimes \mathbb{Z}_3$	$\mathbb{Z}_9 \rtimes \mathbb{Z}_3$
125	$(\mathbb{Z}_5 \times \mathbb{Z}_5) \rtimes \mathbb{Z}_5$	$\mathbb{Z}_{25} \rtimes \mathbb{Z}_5$
343	$(\mathbb{Z}_7 \times \mathbb{Z}_7) \rtimes \mathbb{Z}_7$	$\mathbb{Z}_{49} \rtimes \mathbb{Z}_7$
⋮	⋮	⋮
p^3	$(\mathbb{Z}_p \times \mathbb{Z}_p) \rtimes \mathbb{Z}_p$	$\mathbb{Z}_{p^2} \rtimes \mathbb{Z}_p$

Parte II

PARTE INFORMÁTICA

Las matemáticas son el único material didáctico que se puede presentar de una manera totalmente no dogmática. (Max Dehn)

5

LIBRERÍA EN PYTHON

5.1 INTRODUCCIÓN

Walter Von Dyck es considerado el precursor de la Teoría Combinatoria de Grupos tras sus trabajos sobre la construcción de grupo libre y definición de grupo dado por generadores y relatores. A raíz de sus estudios, en 1911, el matemático Max Dehn publicó un artículo con la formulación de los tres problemas de decisión más conocidos en Teoría de Grupos: El Problema de Palabras, Conjugación e Isomorfismo.

1. El Problema de Conjugación para un grupo es el problema de decisión de determinar si dos elementos $x, y \in G$ son elementos conjugados o no; es decir, si existe $z \in G$ que cumpla $x = zyz^{-1}$. Este problema se conoce también como Problema de la Transformación.
2. El Problema del Isomorfismo para dos grupos es el problema de decidir si son isomorfos o no.
3. El Problema de Palabras para grupos finitamente generados es el problema algorítmico de decidir si dos palabras dadas como producto de generadores representan el mismo elemento. Nos centraremos principalmente en este.

Un año más tarde, en 1912, Dehn dio un algoritmo capaz de resolver el Problema de Conjugación y Palabras para grupos definidos con una única relación. Sin embargo, pasaron años hasta que estos problemas fueron resueltos, siendo muchos los matemáticos quienes poco a poco presentaban soluciones para algunos grupos específicos, sin obtener unos avances importantes. En 1936, J.A. Todd y H.S.M. Coxeter describieron un algoritmo capaz de resolver el Problema de Palabras, el *Algoritmo de Todd Coxeter*, que debido a su complejidad, no se podía ejecutar sobre cualquier grupo.

Así fue hasta la llegada de Alang Turing, el primer matemático e informático que logró formalizar los conceptos de algoritmo y computación. De hecho, es considerado el padre de la computación por todos sus aportes sobre inteligencia artificial, el más importante vino en 1945, el diseño en detalle del primer ordenador.

Con la llegada de los primeros ordenadores electrónicos, muchos matemáticos no dudaron en aprovechar las ventajas que supondrían para sus estudios. Especialmente, los algebraicos vieron como el estudio de nuevos grupos creció inimaginablemente, expandiéndose de una forma nunca antes vista.

A raíz de este origen, se dirigieron muchas investigaciones para promover el estudio de grupos mediante ayuda de ordenadores. Una de las primeras y más importantes fue dirigida en 1951, donde M.H.A. Newman motivó la investigación de todos los grupos de orden hasta 256. En relación con el *Algoritmo de Todd Coxeter*, surgieron las primeras implementaciones a ordenador del algoritmo original y a día de hoy hasta existen algoritmos que pueden resolver el Problema de Palabras en grupos de orden infinito, como es el caso del *Algoritmo de Knuth-Bendix* [KB70].

Nuestro trabajo consistirá en la extensión y optimización de la librería de Teoría de Grupos de José L. Bueso Montero y Pedro A. García Sánchez, basada en la librería de Naftali Harris y disponible en [GSBM16] y [Har12], respectivamente. Esta librería se presenta como un recurso didáctico, complementario y de profundización de la teoría algebraica estudiada durante la carrera. Está disponible en github.com/lmd-ugr/Grupos, y para facilitar su uso, se ha proporcionado un tutorial en Jupyter.

El primer paso realizado fue el estudio teórico de todos los ficheros y métodos programados, donde se anotaron todos los posibles cambios para su modificación posterior. Originalmente, los ficheros eran los siguientes:

- `Set.py`: contiene la clase para definir un conjunto *Set* de tipo *frozenset* (para que éste sea inmutable). Este conjunto contendrá los elementos de un grupo, que tendrán un carácter estático, es decir, no se podrán modificar una vez creados.
- `Function.py`: la clase *Function* simulará la operación binaria del grupo. Su dominio y codominio son conjuntos de tipo *Set* y contiene métodos para comprobar la inyectividad, sobreyectividad y biyectividad de la función.
- `Group.py`: contiene la principal clase de la librería, la clase Grupo. En ella, se le da estructura de grupo a un conjunto de tipo *Set* junto a su operación binaria *Function*. Su constructor se encarga de comprobar que se cumplen los axiomas de grupo (asociatividad, existencia de elemento neutro y existencia de inverso para cada elemento) y contiene métodos que abarcan los principales conceptos de la asignatura Álgebra II.

La librería únicamente permitía la construcción de un grupo dando un conjunto de elementos junto a una operación binaria (o tabla de multiplicar) que satisfagan los axiomas de grupo. Para grupos de orden pequeño esto no supone un gran problema, sin embargo, definir un grupo de orden elevado de esta forma no es recomendable ya que puede llegar a ser muy ineficiente. Por esta razón, se han creado nuevas clases para representar los diferentes grupos, donde se han sobrecargado los métodos que definen cada una de sus operaciones binarias.

Además, se ha realizado la implementación del *Algoritmo de Todd Coxeter*. Así, podremos definir grupos dados en términos de generadores y relatores y obtener su representación por permutaciones, a los que aplicaremos diferentes métodos para identificarlos, mediante isomorfismos, con grupos conocidos. Véase la Sección 2.3.1 para una descripción detallada del algoritmo.

5.2 OPTIMIZACIÓN

En esta sección se comentarán los cambios realizados en cada clase de la librería, que se han implementado siguiendo los conceptos matemáticos descritos anteriormente. Además, se ha completado y añadido la documentación de cada método y función implementada, por lo que el usuario puede consultarla si así lo desea.
Se usará Jupyter para ilustrar algunos ejemplos, donde únicamente bastará con importar la librería que queramos para poder usar todos sus métodos implementados.

- `Set.py`: se han añadido métodos para realizar operaciones a nivel de conjunto.
  - Unión, diferencia, intersección, producto cartesiano y diferencia simétrica.

```
[1]: from Set import Set
```

```
[2]: A = Set({1,2,3})
     B = Set({2,4})
```

```
[3]: C = A*B
     C
```

```
[3]: {(2, 4), (1, 2), (3, 4), (2, 2), (3, 2), (1, 4)}
```

  - *cardinality, is_finite*: se tratan de métodos que sirven para calcular la cardinalidad del conjunto y comprobar si este es finito, respectivamente.

```
[4]: C.is_finite(), C.cardinality()
```

```
[4]: (True, 6)
```

  - *subsets*: este método se encarga de calcular los subconjuntos de un conjunto. Si por parámetro se le pasa un número natural n, entonces calculará los subconjuntos de tamaño n.

```
[5]: A.subsets()
```

```
[5]: [{1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}]
```

```
[6]: A.subsets(2)
```

```
[6]: [{1, 2}, {1, 3}, {2, 3}]
```

- `Function.py`: se ha mantenido en tu totalidad el formato original, a excepción del operador *__str__* que muestra ahora la función de manera clara y precisa.

[7]:
```
from Function import Function
```

Sea S un conjunto de tipo Set, para definir, por ejemplo, la siguiente operación binaria

$$S \times S \to S$$
$$(x, y) \mapsto (x + y)\,\%3$$

usaremos las función lambda que nos ofrece Python.

[8]:
```
S = Set({0,1,2})
F = Function(S*S, S,lambda x: (x[0]+x[1])%3)
print(F)
```

```
f((0, 1))=1
f((1, 2))=0
f((2, 1))=0
f((0, 0))=0
f((1, 1))=2
f((2, 0))=2
f((0, 2))=2
f((2, 2))=1
f((1, 0))=1
```

- `Group.py`:
  - *__str__* y *__repr__*: se modifican para además mostrar los elementos del grupo (siempre que el orden del grupo no sea grande).
  - Se ha modificado el constructor *__init__* de la clase *Group*. De este modo, se podrán definir grupos de las dos formas comentadas en la Sección 2.
    1. Definición axiomatixada. Se comprueba que el par *(Set, Function)* pasado por argumento satisface los axiomas de grupo (asociatividad, identidad e inversos).

[9]:
```
from Group import *
```

[10]:
```
S = Set({0,1,2,3})
F = Function(S*S, S,lambda x: (x[0]+x[1])%4)
Z4 = Group(S,F)

print(Z4)
```

[10]:
```
Group with 4 elements: {0, 1, 2, 3}
```

2. Definición en términos de generadores y relatores. Sea un grupo $G = \langle X \mid R \rangle$. Se pasa por argumento el conjunto de generadores X y relaciones R que definen al grupo. El constructor se encarga de aplicar el *Algoritmo de Todd Coxeter* y darle estructura de grupo de Permutaciones al grupo G. Se tomará el subgrupo trivial para esta ejecución del algoritmo.

   A continuación definimos el grupo $G = \langle a \mid a^4 = 1 \rangle$.

[11]:
```
gens = ['a']
rels = ['aaaa'] #a^4=1

G = Group(gensG=gens, relsG=rels)
print(G)
```

```
Group with 4 elements: {(), (1, 2, 3, 4), (1, 4, 3, 2), (1,␣
↪3)(2, 4)}
```

   Naturalmente, y aunque la forma de definir ambos grupos anteriores es distinta, son isomorfos; es decir, $G = \langle a \mid a^4 = 1 \rangle \cong \mathbb{Z}_4$.

[12]:
```
G.is_isomorphic(Z4)
```

[12]:
```
True
```

   Por último, se añadirá una tercera forma de definir un grupo. Sea Y un conjunto de elementos, entonces el grupo G se definirá como el grupo generado por $\langle Y \rangle$. En el siguiente ejemplo tomaremos un conjunto con una única permutación, sin embargo, no exigimos que los elementos sean permutaciones.

[11]:
```
p = permutation((1,2,3,4))
G = Group(elems=[p])
print(G)
```

```
Group with 4 elements: {(), (1, 2, 3, 4), (1, 4, 3, 2), (1,␣
↪3)(2, 4)}
```

- *is_abelian*: en una primera versión se comprobaba si el grupo era abeliano en el constructor y se hacía uso de una variable de clase. Se añade este método para realizar esta comprobación.
- *identity*: del mismo modo que en *is_abelian*, se añade un nuevo método para calcular la identidad del grupo.

[13]:
```
Z4.is_abelian()
```

[13]:
```
True
```

```
[14]: Z4.identity() , G.identity()
```

```
[14]: (0, ())
```

- *cosets*: método que calcula las clases laterales de un grupo G sobre un subgrupo H. Se optimiza y se simplifica.

- `Permutation.py`: la clase *Permutation* es la que se encarga de construir permutaciones y da lugar al grupo Simétrico y Alternado. En esta clase no se han realizado importantes modificaciones, sin embargo, requiere de una mención especial ya que basándonos en el Teorema 22, se programará un método que le proporcionará una estructura de grupo de Permutaciones a cualquier grupo, en especial, a los grupos definidos por una presentación.

  Las modificaciones realizadas han sido las siguientes:

  - *__mul__*: se modifica el operador encargado de multiplicar dos permutaciones. Optimización y simplificación del código.
  - *__call__*: este operador se encarga de calcular la imagen de un elemento de una permutación. Arrojaba un error de compilación que ya se ha corregido.

```
[15]: p = permutation((1,3),(5,2))

      for i in range(1,6):
          print("p({})={}".format(i, p(i)))
```

```
p(1)=3
p(2)=5
p(3)=1
p(4)=4
p(5)=2
```

  - *even_permutation*, *odd_permutation*: se añaden los siguientes métodos encargados de calcular si una permutación es par o impar.

```
[16]: p.odd_permutation()
```

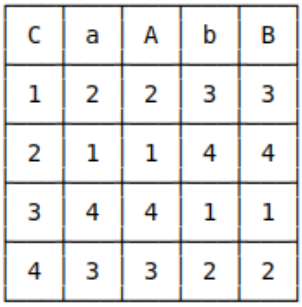
```
[16]: False
```

- `Complex.py`: se ha realizado una implementación de la clase número complejo, *class Complex*, junto a todos los operadores necesarios para realizar operaciones entre números complejos. Gracias a esta clase, se programa el grupo de las raíces n-ésimas de la unidad y una función que se encarga de representar sus soluciones:

  - *plot(G, rep)*: dado un grupo de las raíces n-ésimas de la unidad pasado por argumento, esta función representa todas sus raíces en el plano complejo. El segundo parámetro *rep* permite elegir el modo de representación, que

puede ser "exp" (por defecto) para mostrarlos mediante la representación exponencial o "binom" para mostrarlos usando su forma binomial $a + bi$.

[17]:
```
G = RootsOfUnitGroup(5)
plot(G)
```



C	a	A	b	B
1	2	2	3	3
2	1	1	4	4
3	4	4	1	1
4	3	3	2	2

- `Quaternion.py`: como hemos comentado anteriormente, uno de los problemas que tenía la librería y que se quería corregir era evitar tener que dar la tabla de multiplicar de un grupo. Por ello, se realiza una implementación de los números cuaternios en la clase *Quaternion* sobrecargando el operador *__mul__* para dotar a estos números de su producto.
  - Se han programado todos los operadores necesarios para trabajar y operar con números cuaternios, desde su manejo y representación *__repr__*, *__str__*, *__call__*, hasta los operadores encargados de sumar, restar, multiplicar, dividir (*__add__*, *__sub__*, *__mull__*, *__div__*)... etc.
  - Se han implementado métodos como *conjugate*, *norm*, *inverse*, *trace*, encargados de calcular el conjugado, norma, inverso y traza, respectivamente.

[18]:
```
q = Quaternion(-3,1,2,-8)
p = Quaternion(0,2,3,1)
q+p
```

[18]:
```
-3+3i+5j-7k
```

[19]:
```
q*p
```

[19]:
```
20i-26j-4k
```

[20]:
```
(q*p).conjugate() + 2*(q-3*p)
```

[20]:
```
-6-30i+12j-18k
```

[21]:
```
i = Quaternion(0,1,0,0)
j = Quaternion(0,0,1,0)
k = Quaternion(0,0,0,1)
```

[22]:
```
i*i == j*j == k*k == i*j*k == -1
```

```
[22]: True
```

La función que se encarga de crear el grupo de los Cuaternios es *QuaternionGroup*, donde únicamente se le ha de pasar por argumento una de las dos representaciones siguientes:

```
[23]: Q = QuaternionGroup(rep="ijk")
      print(Q)
```

```
[23]: Group with 8 elements: { 1,  i,  j,  k,  -k,  -j,  -i,  -1}
```

```
[24]: Q2 = QuaternionGroup(rep="permutations")
      print(Q2)
```

```
Group with 8 elements: {(1, 4, 3, 2)(5, 7, 8, 6), (1, 7, 3,␣
↪6)(2, 8, 4, 5), (1, 6, 3, 7)(2, 5, 4, 8), (1, 8, 3, 5)(2, 6,␣
↪4, 7), (1, 2, 3, 4)(5, 6, 8, 7), (1, 5, 3, 8)(2, 7, 4, 6),␣
↪(), (1, 3)(2, 4)(5, 8)(6, 7)}
```

```
[25]: Q.is_isomorphic(Q2)
```

```
[25]: True
```

- Por último, se ha programado la función *QuaternionGroupGeneralised(n)* que define el grupo generalizado de los Cuaternios, con presentación:

$$Q_n = \langle a, b \mid a^n = b^2, a^{2n} = 1, b^{-1}ab = a^{-1} \rangle .$$

Cuando $n = 2$ se tiene el grupo de los Cuaternios.

```
[26]: Q3 = QuaternionGroupGeneralised(2)
      Q3.is_isomorphic(Q)
```

```
[26]: True
```

- `Dihedral.py`: del mismo modo que en el grupo de los Cuaternios, se ha realizado la implementación del grupo Diédrico en la clase Dihedral, *class dihedral*. Ahora, un grupo Diédrico D_n, de orden $2n$, almacenará internamente n simetrías y n rotaciones que podrán representarse de tres formas equivalentes:

  1. *RS*: el conjunto de rotaciones serán denotadas por $R0, R1, \ldots, RN$ y las simetrías por $S1, S2, \ldots, SN$.
  2. *Permutations*: se representará el grupo como un grupo de permutationes.
  3. *Matrix*: se trata de una representación que hace referencia a la matriz del movimiento asociado.

[27]:
```
D4 = Dihedral(4)
print(D4)
```

```
Rotaciones:                  Reflexiones:
(1.0, -0.0, 0.0, 1.0)       (1.0, 0.0, 0.0, -1.0)
(0.0, -1.0, 1.0, 0.0)       (0.0, 1.0, 1.0, -0.0)
(-1.0, -0.0, 0.0, -1.0)      (-1.0, 0.0, 0.0, 1.0)
(-0.0, 1.0, -1.0, -0.0)      (-0.0, -1.0, -1.0, 0.0)
```

Para construir el grupo basta con llamar a la función *DihedralGroup*. El primer parámetro n hace referencia al grupo que se desea crear, que tendrá orden $2n$, mientras que el segundo parámetro sirve para indicar la representación deseada.

[28]:
```
D = DihedralGroup(3, rep='RS')
print(D)
```

```
Group with 6 elements: {'R1', 'R0', 'R2', 'S2', 'S1', 'S0'}
```

Comprobemos que efectivamente los grupos son isomorfos aunque su representación sea distinta:

[29]:
```
D2 = DihedralGroup(3, rep="matrix")
D3 = DihedralGroup(3, rep="permutations")
```

[30]:
```
D.is_isomorphic(D2), D2.is_isomorphic(D3)
```

[30]:
```
(True, True)
```

5.3 ALGORITMO DE TODD COXETER

Sea $G = \langle X \mid R \rangle$ un grupo dado por generadores y relatores. A raíz de lo explicado en la Sección de Matemáticas sobre las presentaciones, nos encontramos ante el problema de desconocimiento de la estructura del grupo. Existen algunos grupos definidos por una presentación que son sencillos de identificar estableciendo isomorfismos y usando el *Teorema de Dyck* 15. En términos computacionales, se podrían plantear algoritmos que operen a partir de los generadores del grupo y obtengan todos los elementos que satisfagan las relaciones dadas, sin embargo, en el caso general esto no va a ser posible.

Inicialmente, no se sabe el orden del grupo y, por ello, no tener un criterio de parada definido en un algoritmo no es una buena técnica. En términos de eficiencia tampoco ya que: ¿Cuándo debe detenerse?, ¿Cuánto tiempo deberá estar calculando elementos?, ¿Qué ocurre si el algoritmo cicla sin obtener todos ellos?, ¿Cómo sabemos si dos palabras son el mismo elemento (*Word Problem*)?

Para responder a todas estas preguntas se utilizará el *Algoritmo de Todd Coxeter*, un método matemático que resuelve el Problema de Palabra mediante la enumeración de clases. Este algoritmo se considera uno de los métodos fundamentales de la Teoría de Grupos ya que a partir de un grupo G definido por una presentación y un subgrupo $H \leq G$, es capaz de resolver el Problema de Palabras enumerando las clases laterales de H en G. Véase [TC36] para la descripción del procedimiento original y [Mil16] para una explicación actualizada del mismo.

5.3.1 *Implementación*

Aunque originalmente el *Algoritmo de Todd Coxeter* tenía una única versión, se han desarrollado diferentes implementaciones que se diferencian principalmente por la estrategia de realizar definiciones y por el nivel de cómputo y *CPU* usados. En particular, el método usado es conocido por *HLT* , desarrollado por *Hazelgrove, Leech y Trotter*, descrito en [DBO06], por lo que el lector puede consultarlo para una descripción más teórica. Basándonos en esta misma documentación, desarrollaremos los diferentes métodos en relación con este algoritmo que se han implementado.

Sea G un grupo finitamente presentado y H un subgrupo de G. El principal objetivo del algoritmo es verificar si el índice $[G : H]$ es finito, por lo que nos encontramos en las siguientes situaciones:

1. Si el algoritmo termina, entonces $[G : H]$ es finito y coincidirá con el número de clases laterales de G sobre H; como consecuencia, se obtendrá una tabla completa de clases laterales.
2. Si el algoritmo no termina en un tiempo finito entonces puede ser que la presentación dada pertenezca a un grupo de orden muy alto, que no sea compatible con la versión HLT del algoritmo o que el índice $[G : H]$ sea infinito, lo que significa que el orden de G es infinito. En este caso, en teoría se debe ejecutar ininterrumpidamente, sin embargo, en la práctica se queda sin espacio.

Comentario. Debido al apartado anterior (2), muchos autores consideran que el término "algoritmo" atribuido a este procedimiento no es correcto.

La entrada del algoritmo consiste en un grupo finitamente presentado $G = \langle X, R \rangle$ y un subgrupo $H = \langle Y \rangle$. Definimos $A := X \cup X^{-1}$. El conjunto de relatores R de G están dados como palabras en A. Además, el conjunto de generadores de H, es dado también como un subconjunto de palabras de A. Asumimos que tanto R como Y son palabras reducidas.

En nuestra implementación, los generadores del grupo serán las letras $(a, b, c, \dots)$, mientras que sus inversos $(a^{-1}, b^{-1}, c^{-1}, \dots)$ serán representados por letras mayúsculas $(A,B,C, \dots)$.

Como se vio en la Sección 2.3.1.1, la tabla de clases laterales asociada al grupo G es equivalente al grafo de Schreier que refleja la acción a derecha de G sobre G/H. En este grafo, cada vértice enumerado desde $1, 2, \dots, n$ hace referencia a una clase lateral de G/H. La tabla de clases será definida como una quíntupla $C := (\tau, \chi, p, n, M)$, donde:

- p es la aplicación $p : [1, \dots, n] \to [1, \dots, n]$. Esta aplicación hace referencia a las clases de equivalencia, donde cada una estará representada por su menor elemento. Así, se debe cumplir que $p(\alpha) \leq \alpha$, dándose la igualdad si α es el representante de su clase de equivalencia.

  Definimos el conjunto de *clases vivas* como:

  $$\Omega = \{\alpha \in [1, \dots, n] \text{ tal que } p(\alpha) = \alpha\}.$$

  Ante una nueva definición de una clase α, se debe cumplir que $p(\alpha) = \alpha$. Sin embargo, durante el trascurso del procedimiento se puede dar el caso de que dos clases α y β con $\alpha < \beta$ representen la misma clase lateral. Esta rutina se llama coincidencia y en la Sección 5.3.1.2 se verá como procesarlas.

- τ es una aplicación $\tau : [1, \dots, n] \to A^*$. La palabra $\tau(\alpha)$ es un representante de la clase correspondiente a $\alpha \in \Omega$. No es necesario almacenar estos representantes en la implementación del algoritmo, sin embargo, son importantes para la descripción teórica.

  La clase 1 pertenece a Ω y se cumple que $\tau(1) = \epsilon$. Por tanto, $H\tau(\epsilon) = H$, que pertenece a G/H. Evaluando los valores $2, \dots, n$ en τ, se obtendrán el resto de elementos $H\tau(2), \dots, H\tau(n)$ (clases laterales derechas 5), que pertenecen también a G/H.

- χ es la aplicación $\chi : [1, \dots, n] \times A \to [1, \dots, n]$. Se usará una matriz para su implementación, donde las diferentes clases $[1, \dots, n]$ se dispondrán en la

primera columna, los valores de A (generadores e inversos) en la cabecera y cada entrada de la matriz tomará un valor natural menor o igual que n.

C	a_1	a_2	$\dots$	a_r
1	$\chi(1,a_1)$	$\chi(1,a_2)$	$\dots$	$\chi(1,a_r)$
2	$\chi(2,a_1)$	$\chi(2,a_2)$	$\dots$	$\chi(2,a_r)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$\chi(n,a_1)$	$\chi(n,a_2)$	$\dots$	$\chi(n,a_r)$

La tabla de clases C se dice que está *completa* si no tiene entradas sin definir en las clases que aún están vivas, es decir, $\chi(\alpha,x)$ está definida para todo $\alpha \in \Omega$, $x \in A$. A partir de aquí, se usará notación exponencial α^x para representar $\chi(\alpha,x)$.

- $n, M \in \mathbb{N}$ con $1 \leq n \leq M$ donde M es un valor fijo que representa el mayor número de clases *permitidas*, es decir, determina la cantidad máxima de memoria que el algoritmo puede usar. En nuestra implementación se ha fijado a $1E6$, y en todos los ejemplos probados, el algoritmo termina. Por otro lado, n representa el mayor número que se ha usado para una clase *viva*.

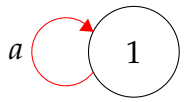
Para familiarizarnos con esta notación, realizamos a continuación un ejemplo.

Consideremos el grupo:

$$G = \langle a, b \mid a^3, b^3, aba^{-1}b^{-1} \rangle.$$

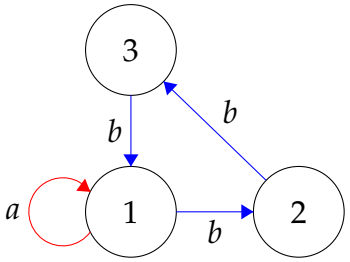
Sea $H = \langle a \rangle \leq G$. Es claro que G es producto directo de dos grupos cíclicos de orden 3, y que $[G : H] = 3$. Veamos que es correcto usando el *Algoritmo de Todd Coxeter*.

Comenzamos representando con el número 1 la clase trivial de H en G, es decir, H. Como $a \in H$, se tiene que $Ha = H$. En primer lugar, todos los generadores de H deben satisfacerse para la clase 1, por ello, definimos $1^a := 1$.



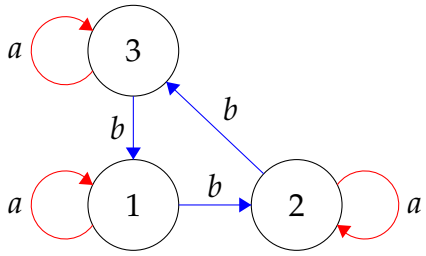
Ahora, realizamos un proceso conocido como escaneo de relatores bajo las clases. Como cada relator w representa la identidad cuando es visto como elemento de G, se debe cumplir $\alpha^w = \alpha$, para toda clase $\alpha \in \Omega$ y toda palabra $w \in R$. En términos del grafo de Schreier, se debe poder realizar un recorrido (marcado por cada relator) partiendo y terminando en el mismo vértice. Esto se cumple para $1^{a^3} = 1$, pero no para el resto, por lo que el escaneo se interrumpe. Por esta razón, se deben realizar nuevas definiciones para que el proceso de escaneo se complete para todas las clases.

El siguiente relator que se debe escanear es b^3 sobre la clase 1, que no se cumple. Como no se puede asegurar $1^b = 1$, necesitamos definir nuevas clases. $1^b := 2$, $2^b := 3$ y $3^b := 1$. Equivalentemente, estas definiciones equivalen a $1^{b^{-1}} = 3$, $2^{b^{-1}} = 1$ y $3^{b^{-1}} = 2$.



Como podemos comprobar, el vértice 1 cumple $1^{a^3} = 1$ y $1^{b^3} = 1$; los vértices 2 y 3 cumplen $2^{b^3} = 1$ y $3^{b^3} = 1$, respectivamente, pero aún falta cumplir la relación b^3 sobre los vértices 2 y 3, y $aba^{-1}b^{-1}$ sobre todos los vértices.

Para hacer cumplir $\alpha^w = \alpha$ para los vértices $2, 3$ y todas las relaciones, se define $2^a := 2$ y $3^a := 3$. Estas dos definiciones bastan para terminar el escaneo en todos los vértices y todo relator ya que el tercer relator termina satisfaciéndose en todo vértice, obteniendo el siguiente grafo de Schreier:



El vértice 1 equivale a la clase $H\tau(1) = H$, el vértice 2 a la clase $H\tau(2) = Hb$ y, el 3 a la clase $H\tau(3) = Hb^{-1}$. Este grafo de Schreier es equivalente a la siguiente tabla de clases:

	a	a^{-1}	b	b^{-1}
1	1	1	2	3
2	2	2	3	1
3	3	3	1	2

Como todas las clases se escanean correctamente bajo todos los relatores, y todas las entradas en la tabla de clases están definidas, el proceso termina, y el índice de H en G, $[G : H]$, coincide con el número de clases o vértices definidos, que son 3.

A continuación, enunciaremos unas **propiedades** que se deben cumplir, y en el Teorema 22, se probará que el algoritmo es correcto y que los generadores de Schreier dan lugar a una representación por permutaciones del grupo G.

1. $1 \in \Omega$ y $\tau(1) = \epsilon$.
2. $\alpha^x = \beta \iff \beta^{x^{-1}} = \alpha$.
3. Si $\alpha^x = \beta$, entonces $H\tau(\alpha)x = H\tau(\beta)$.
4. Para todo $\alpha \in \Omega$, $1^{\tau(\alpha)}$ está definido y es igual a α.

Teorema 22. *Supongamos que:*

(I) *Las propiedades* $1-4$ *anteriores se cumplen.*

(II) *La tabla de clases está completa.*

(III) 1 *escanea satisfactoriamente para todo* $w \in Y$.

(IV) *Todo* $\alpha \in \Omega$ *se escanea correctamente para todo* $w \in R$.

Entonces, $[G : H] = |\Omega|$. *Además, para cada* $x \in A$, *la aplicación:*

$$\begin{aligned} \varphi(x) \colon \Omega &\to \Omega \\ \alpha &\mapsto \alpha^x \end{aligned}$$

es una permutación de Ω *y* φ *extiende a un homomorfismo de* $G = \langle X \mid R \rangle$ *a* $S(\Omega)$, *que es equivalente a la acción de* G *sobre las clases de* H *por la multiplicación a la derecha.*

Demostración. Por la propiedad 2, $\varphi(x)$ y $\varphi(x^{-1})$ son aplicaciones inversas para cualquier $x \in A$ y entonces, $\varphi(x)$ es una permutación. La suposición (iv) dice que para cualquier $w = x_1 \dots x_r \in R$, se tiene $\alpha^{\varphi(x_1)\varphi(x_2)\dots\varphi(x_r)} = \alpha$, para todo $\alpha \in \Omega$. Así, $\varphi(x_1)\varphi(x_2)\cdots\varphi(x_r) = 1_{S(\Omega)}$. Por lo que φ extiende a un homomorfismo de grupos, que viene dado por $\varphi^* \colon \langle X \mid R \rangle \to S(\Omega)$; $\varphi^*(xN) = \varphi(x)$, para todo $x \in X$, donde $N = \ker(\varphi^*)$.

Para probar la equivalencia de φ con la acción de G sobre el conjunto C de clases de H, definimos:

$$\begin{aligned} \overline{\tau} \colon \Omega &\to C \\ \overline{\tau}(\alpha) &= H\tau(\alpha) \end{aligned}$$

Para cualquier $v \in A^*$, la suposición (ii) y la propiedad 3 implican que $\overline{\tau}(1^v) = Hv$, por lo que $\overline{\tau}$ es sobreyectiva. Si $\overline{\tau}(\alpha) = \overline{\tau}(\beta)$, entonces $H\tau(\alpha) = H\tau(\beta)$, por lo que $\tau(\alpha)\tau(\beta)^{-1} \in H$. Así, $\tau(\alpha)\tau(\beta)^{-1}$ es el producto $w_1 \cdots w_r$, donde $w_i \in Y \cup Y^{-1}$.
Por la hipótesis (iii), se tiene $1^{w_i} = 1$ para todo w_i. Así, $1^{\tau(\alpha)\tau(\beta)^{-1}} = 1$, y por la propiedad 4, $\alpha = 1^{\tau(\alpha)} = 1^{\tau(\beta)} = \beta$. Se tiene que $\overline{\tau}$ es inyectiva, y por tanto, biyectiva, luego $[G : H] = |\Omega|$ y, por la propiedad 3, $\overline{\tau}$ define una equivalencia entre φ y la acción de G sobre C por la multiplicación a la derecha. □

5.3.1.1 *Proceso definición de clases*

Si α^x no está definido para algún $\alpha \in [1,\dots,n]$, $x \in A$, la manera más simple es añadir un nuevo elemento β a Ω y definir $\alpha^x := \beta$. Esto es lo que se conoce como *definición de una clase* y el pseudocódigo es el que se muestra a continuación:

Algorithm 1: Define

Data: $\alpha \in \Omega$, $x \in A$, α^x undefined
Result: C: Coset Table with new α^x definition
if $n = M$ **then**
  abort
end
$n := n+1, \quad \beta := n, \quad p(\beta) := \beta$
$\alpha^x := \beta\ , \quad \beta^{x^{-1}} := \alpha$

Como podemos observar, el proceso previo al de la definición de la clase, es el de comprobar si hay espacio disponible, es decir, n debe ser menor que M. Además, cuando se llama al procedimiento *Define* anterior para definir $\alpha^x := \beta$, también se debe definir $\beta^{x^{-1}} := \alpha$ por la propiedad 2 anterior.

Como consecuencia del proceso de definición de una clase, se han programado los siguientes métodos:

- *isAlived(α)*: comprueba que la clase pasada por argumento está viva, es decir, pertenece a Ω.
- *Undefine(α, x)*: se trata de la operación opuesta a `define` y su función será la de eliminar el valor α^x para un $\alpha \in \Omega$ y $x \in A$ de la tabla C.
- *isDefined(α, x)*: se encarga de comprobar si el valor α^x para un $\alpha \in \Omega$ y $x \in A$ está definido, devolviendo *True* en caso afirmativo.

5.3.1.2 *Coincidencias*

En nuestro algoritmo, para almacenar la relación de equivalencia de una clase se ha programado la clase *EquivalenceClass*. En ella, se lleva un registro de las relaciones de equivalencia, donde cada una estará representada por su menor elemento. Destacamos los siguientes métodos:

- *rep(k)*: devuelve el menor elemento (su representante) de la clase de equivalencia de la clase k. Este método hace uso de la aplicación $p : [1,\dots,n] \to [1,\dots,n]$ descrita anteriormente.
- *merge(k,λ)*: se opera sobre las dos clases de equivalencia pasadas por argumento, devolviendo su representante o -1 si las dos clases son iguales.

Como ya hemos comentado, una coincidencia ocurre cuando dos clases distintas α y β con $\alpha < \beta$ representan la misma clase, es decir, tienen el mismo representante. Se debe registrar esta ocurrencia llamando al método $merge(\alpha, \beta)$ anterior, el cual fija $p(\beta) = \alpha$, para indicar que β es equivalente a α, e introducir β en una cola (queue) de clases

que deben ser procesadas. $p(\alpha) = \alpha$ se debe mantener ya que α es el representante de ambas clases y debe perdurar en Ω.

El pseudocódigo asociado a este método es el siguiente:

Algorithm 2: Coincidence

```
Data: α, β ∈ Ω
merge(α, β)
while |queue| > 0 do
    y := queue.front(),    queue.pop()
    for x ∈ A do
        if y^x isDefined then
            ρ = rep(y^x)
            undefine ρ^(x^-1)
            μ := rep(y),    σ := rep(ρ)
            if μ^x isDefined then
                merge(σ, μ^x)
            else if σ^(x^-1) isDefined then
                merge(μ, σ^(x^-1))
            else
                μ^x := σ ,    σ^(x^-1) := μ
            end
        end
    end
end
```

Sea y un elemento de la cola que debe ser procesado. Toda la información de la clase y debe ser transferida a la clase $rep(y)$ ya que va a ser borrada. Para cada elemento $x \in A$, se debe comprobar si y^x está definido, supongamos que $y^x = \rho$. Entonces, lo primero se debe llevar a cabo es borrar esta entrada ya que no queremos que y permanezca en la fila ρ. De este modo, queda eliminada la clase y de la tabla. En términos del grafo de Schreier, se han borrado todas las aristas que llegan al vértice y. Sin embargo, debemos completar el grafo con nuevas aristas que entran y salen de su representante $rep(y)$.

Consideramos $\overline{y} = rep(y)$ y $\overline{\rho} = rep(\rho)$. Distingamos los siguientes casos:

1. Si se tiene una entrada definida $\overline{y}^x$, que es igual a w, entonces llamamos a $merge(\overline{\rho}, w)$.
2. Si se tiene una entrada definida $\overline{\rho}^{x^{-1}} = w$, entonces realizamos un $merge(\overline{y}, w)$.
3. En caso contrario, se tendrán que definir las entradas $\overline{y}^x := \overline{\rho}$ y $\overline{\rho}^{x^{-1}} := \overline{y}$.

5.3.1.3 *Escaneo*

Sigamos ahora con el *proceso de escaneo*. Este procedimiento recibe una clase $\alpha \in \Omega$ y una palabra $w \in A$ y se encarga de comprobar si la palabra w se satisface para la clase α, es decir, $\alpha^w = \alpha$. En otras palabras, si partiendo desde el vértice α se puede realizar un recorrido marcado por w que acabe en el mismo vértice.

El método encargado de comprobar lo anterior se denomina *ScanAndFill* y el pseudocódigo asociado es el siguiente:

Algorithm 3: ScanAndFill

Data: $\alpha \in \Omega$, $w = x_1, \ldots, x_r$ with $x_i \in A$
$i, j := 0, r\ , \quad f, b := \alpha, \alpha$
while *True* **do**
  Scan forward
  while $i \leq j$ *and* f^{x_i} *isDefined* **do**
    $f := f^{x_i}\ , \quad i := i + 1$
  end
  if $i > j$ **then**
    if $f \neq b$ **then**
      $Coincidence(f, b)$
    end
    return
  end
  Scan backwards
  while $j \geq i$ *and* $b^{x_i^{-1}}$ *isDefined* **do**
    $b := b^{x_i^{-1}}, \quad j := j - 1$
  end
  if $j < i$ **then**
    $Coincidence(f, b)$
    return
  else if $j = i$ **then**
    $f^{x_i} := b\ , \quad b^{x_i^{-1}} := f$
    return
  else
    $define(f, x_i)$
  end
end

En método se ejecuta ininterrumpidamente hasta que la fila de la clase $\alpha \in \Omega$ se rellene por completo. Se puede realizar el proceso de escaneo hacia adelante o hacia atrás, al igual que se hizo en la Sección 2.3.1 con las tablas de relatores. Cuando el escaneo es satisfactorio salimos del método con una llamada a return. Antes de esta llamada pueden ocurrir dos situaciones, que se detecte una deducción o una coincidencia entre dos clases. En este último caso, se debe llamar al método *Coincidence* para procesarlas. Por último, cuando el método no es capaz de escanear la palabra w

sobre la clase α de forma satisfactoria, se debe definir una nueva clase y seguir hasta que así lo sea.

5.3.1.4 *Método principal*

Explicaremos a continuación la idea que sigue el método *HLT*:

1. En primer lugar, se inicializa una tabla de clases C vacía para el grupo G dado por generadores X y relatores R.
2. Para cada uno de los generadores de H, se llama al método *ScanAndFill* para realizar un escaneo completo sobre la primera clase 1.
3. Se recorre el conjunto de clases vivas Ω y relatores de G, comprobando que cada clase de Ω se escanee por completo siguiendo cada $w \in R$, es decir, $\alpha^w = \alpha$ para todo $\alpha \in \Omega$, $w \in R$. Si no es así, se realizarán sucesivas definiciones para que el escaneo se complete o se alcance la cota M establecida.

El pseudocódigo asociado al método principal *HLT* es el siguiente:

Algorithm 4: CosetEnumeration

```
Data: G = ⟨X | R⟩ : group, H = ⟨Y⟩ : subgroup.
Result: C: Coset Table for G/H.
Initialize an empty Coset table for ⟨X | R⟩.
for w ∈ Y do
 | ScanAndFill(1, w)
end
for α ∈ Ω do
 | for w ∈ R do
 | | if isAlive(α) then
 | | | ScanAndFill(α, w)
 | | end
 | end
 | if isAlive(α) then
 | | for x ∈ A do
 | | | if not isDefined(α,x) then
 | | | | define(α, x)
 | | | end
 | | end
 | end
end
```

Hay que tener cuidado con el bucle que opera sobre las clases de Ω ya que este conjunto no está fijo, es decir, inicialmente contiene la clase 1 pero se van añadiendo nuevas cada vez que sea necesario.

Como hemos comentado anteriormente, uno de los problemas del *Algoritmo de Todd Coxeter* es el gran uso de memoria que requiere. El método descrito anteriormente no es siempre óptimo en términos del máximo valor $max|\Omega|$ escogido, pero en la

mayoría de las veces cumple su función correctamente, proporcionando así suficiente memoria para que el algoritmo termine. En nuestra implementación, el valor $M = max|\Omega| = 1E6$ se ha elegido tras varias ejecuciones en diferentes ejemplos de grupos, por lo que en principio, el espacio no será un problema.

5.3.2 *Funcionalidades y uso*

El *Algoritmo de Todd Coxeter* se ha programado en *ToddCoxeter.py*. Destacamos los siguientes métodos implementados:

- *readGroup*: implementación de una función que nos ayudará a leer los grupos por ficheros. Por orden, se leerán los generadores del grupo G, sus relaciones y los generadores del subgrupo H. En el directorio /Group se proporcionaran ejemplos de diferentes grupos, tanto de aquellos con los que se han trabajado como los que no.
- *CosetEnumeration*: método principal para llamar al algoritmo y generar la tabla de clases laterales de G/H.
- *coset_table* , *schreier_graph*: el primer método devuelve la tabla de clases laterales de G/H, donde el número de filas coincide con el índice $[G : H]$ (sin contar la cabecera). El segundo método calcula el grafo de Schreier resultante, que es equivalente a la tabla de clases anterior.

  Sean w_1, w_2 dos palabras dadas como producto de generadores. Se puede comprobar de forma sencilla si representan el mismo elemento. Para ello, se debe partir del vértice 1, seguir el recorrido de la palabra en el grafo y ver si acaban en el mismo vértice. Por esta razón, el *Algoritmo de Todd Coxeter* es un algoritmo que resuelve el Problema de Palabras.
- *usedCosets*, *FinalCosets*: el primer método devuelve el número de clases usadas durante la ejecución del algoritmo mientras el segundo devuelve el número de clases vivas, que coincidirán con el índice de $[G : H]$ si el algoritmo termina. Si H es el subgrupo trivial entonces este método devuelve el orden de G.
- *getGenerators*: como se ha visto en el Teorema 22, se puede obtener una representación por permutaciones del grupo G. Esta función se encarga de obtener los generadores de Schreier, que más adelante pueden ser usados en el constructor de la clase Group para definir el grupo que generan.

En primer lugar, importamos las librerías que se utilizarán:

- `Group`: fichero principal de la librería. En él se encuentran las principales clases y funciones para definir los diferentes grupos y sirve para identificar y dar estructura de grupo al conjunto de generadores y relatores dados como entrada.
- `ToddCoxeter`: contiene la implementación del *Algoritmo de Todd Coxeter* junto a todas las funcionalidades descritas anteriormente.

```
from Group import *
from ToddCoxeter import CosetTable, readGroup
```

El procedimiento a seguir se desarrollará a continuación y se ilustrará con el siguiente ejemplo sencillo:

$$G = \langle a, b \mid a^2, b^2, ab = ba \rangle \quad y \quad H = \{1\}.$$

1. Leemos los datos de entrada, ya sea mediante variables para definir el grupo o haciendo uso del método *readGroup*, en el que se le ha de especificar la ruta del fichero donde está el grupo.

```
[1]: gen = ['a','b']
     rels = ['aa','bb','abAB']
     genH = []
```

2. Creamos un objeto de la clase *CosetTable*. Después, llamamos al método *CosetEnumeration* para aplicar el *Algoritmo de Todd Coxeter*.

```
[2]: G = CosetTable(gen,rels, genH)
     G.CosetEnumeration()
```

Podemos mostrar la tabla de clases laterales de G/H, que se obtiene mediante el método *coset_table*.

```
[3]: T = G.coset_table()
     print(T)
```

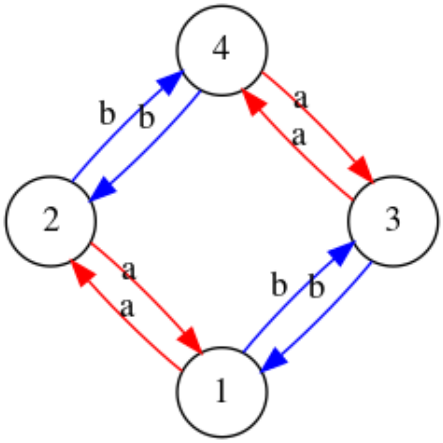
C	a	A	b	B
1	2	2	3	3
2	1	1	4	4
3	4	4	1	1
4	3	3	2	2

Como hemos comentado anteriormente, el número de filas sin contar las cabeceras indica el número de elementos del grupo G/H, que coincide con el índice $[G : H]$. En este ejemplo, se tiene:

$$[G : H] = \frac{|G|}{|H|} = \frac{|G|}{1} = |G| = 4\,.$$

Equivalentemente, el grafo de Schreier asociado se llama de la siguiente forma:

[4]:
```
G.schreier_graph(notes=False)
```



3. El siguiente paso es el de obtener los generadores de Schreier y definir el grupo como aquel generado por estos elementos.

[5]:
```
def print_gens(gens):
    for i in range(len(gens)):
        print(f"g{i} = {gens[i]}")

generators = G.getGenerators()
print_gens(generators)
```

[5]:
```
g0 = (1, 2)(3, 4)
g1 = (1, 3)(2, 4)
```

En este momento, tenemos los generadores que definen al grupo. En particular, en este ejemplo el grupo estará definido por $G = \langle g0, g1 \rangle$.

[6]:
```
Gr = Group(elems=generators)
```

4. Usar el método *is_isomorphic* para identificar cada grupo con grupos conocidos.

   Observando la presentación dada, no es dificil ver que el grupo debe ser isomorfo al Grupo de Klein:

[7]:
```
K = KleinGroup()
Gr.is_isomorphic(K)
```

[7]:
```
True
```

5.3.3 *Ejemplos*

En la siguiente sección se estudiarán diferentes ejemplos de grupos finitamente presentados. El objetivo es identificar la presentación dada estableciendo isomorfismos con grupos estudiados.

Consideramos el siguiente grupo, que se encuentra en *Groups/1.txt*.

$$G = \langle a, b \mid ab^{-1}b^{-1}a^{-1}bbb, b^{-1}a^{-1}a^{-1}baaa \rangle \quad y \quad H = \{1\}.$$

En primer lugar, usamos el método *readGroup* para leerlo:

```
[1]: file = "Groups/1.txt"
     f = readGroup(file)
     print(f)
```

```
(['a', 'b'], ['aBBAbbb', 'BAAbaaa'], [])
```

De igual modo que en el ejemplo anterior, aplicamos el *Algoritmo de Todd Coxeter*.

```
[2]: G = CosetTable(file)
     G.CosetEnumeration()
```

A continuación, mostramos la tabla de clases de G/H:

```
[3]: print(G.table)
```

C	a	A	b	B
1	1	1	1	1

Como vemos, únicamente hay una fila de clases, por lo que $[G : H] = 1$ y, como $H = \{1\}$, G debe ser necesariamente el grupo trivial. Como consecuencia, el grafo de Schreier posee un único vértice. En este ejemplo, no es de utilidad darle estructura de grupo a G, en cambio, servirá para mostrar un ejemplo de un grupo sencillo en el que el número de clases que se usan es muy elevado.

```
[4]: u = G.usedCosets()
     f = G.finalCosets()

     print("Clases usadas: {} \n Clases vivas: {}".format(u,f))
```

```
[8]: Clases usadas: 85
     Clases vivas: 1
```

El siguiente grupo G se encuentra definido en *Groups/S3_2.txt*, y como subgrupo $H \leq G$, tomamos aquel generado por a.

$$G = \langle a, b \mid a^2 = b^2 = 1, (ab)^3 = 1 \rangle \quad y \quad H = \langle a \rangle.$$

[1]:
```
file = "Groups/S3_2.txt"
f = readGroup(file)
f
```

[2]:
```
(['a', 'b'], ['aa', 'bb', 'ababab'], ['a'])
```

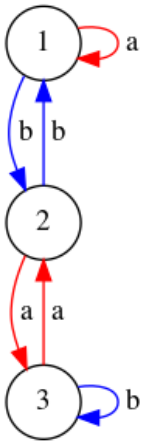
Mostramos la tabla de clases de G/H resultante tras aplicar el algoritmo.

[3]:
```
print(G.coset_table())
```

C	a	A	b	B
1	1	1	2	2
2	3	3	1	1
3	2	2	3	3

Ahora, el grafo de Schreier equivalente:

[4]:
```
G.schreier_graph(notes=False)
```



Obtenemos los generadores de Schreier que definen al grupo, que se usarán, por el Teorema 22, para darle estructura de grupo de Permutaciones.

[5]:
```
generators = G.getGenerators()
print_gens(generators)

S = Group(elems=generators)
```

[5]:
```
g0 = (2, 3)
g1 = (1, 2)
```

Vemos que el grupo es isomorfo a S_3, el grupo Simétrico de orden 6.

[6]:
```
S3 = SymmetricGroup(3)
S3.is_isomorphic(S)
```

[6]:
```
True
```

Otra forma equivalente para construir el grupo S_3 es mediante el producto semidirecto $A_3 \rtimes K$, donde $K = \{1, (12)\}$. Véase el Ejemplo 8.

[7]:
```
A = AlternatingGroup(3)
B = SymmetricGroup(2)

print(A, B)
```

[7]:
```
Group with 3 elements: {(1, 2, 3), (), (1, 3, 2)}
Group with 2 elements: {(), (1, 2)}
```

Para construir este producto semidirecto, habrá que ver las posibles acciones:

$$\varphi \colon K \to \mathrm{Aut}(A_3)$$

[8]:
```
AutA = A.AutomorphismGroup()
Hom = B.AllHomomorphisms(AutA)
```

[9]:
```
hom0 = GroupHomomorphism(B, AutA, lambda x:Hom[0](x))
hom1 = GroupHomomorphism(B, AutA, lambda x:Hom[1](x))

SP0 = A.semidirect_product(B,hom0) #hom0 es la acción trivial
SP1 = A.semidirect_product(B,hom1) #Acción NO trivial
```

Por el Comentario 4.2, hay dos posibles productos semidirectos que vendrán determinados por la acción tomada. Si la acción es la trivial, el producto semidirecto coincidirá con el producto directo y será el grupo cíclico $\mathbb{Z}_{2p}$.

[10]:
```
A.direct_product(B).is_isomorphic(SP0)
```

[10]:
```
True
```

Mientras que si consideramos la acción no trivial obtendremos un producto semidirecto que será isomorfo al grupo D_p. Para $n = 3$, se tiene que $D_3 \cong S_3$ por lo que terminamos la construcción de S_3 como producto semidirecto viendo que $S_3 \cong A \rtimes_{hom1} B$.

[11]:
```
SP1.is_isomorphic(S)
```

[11]:
```
True
```

Consideramos el siguiente ejemplo, disponible en *Groups/D4.txt*.

$$G = \langle a, b \mid a^4, b^2, bab^{-1}a \rangle \quad y \quad H = \langle b \rangle.$$

[1]:
```
file = "Groups/D4.txt"
f = readGroup(file)
f
```

[1]:
```
(['a', 'b'], ['aaaa', 'bb', 'baBa'], ['b'])
```

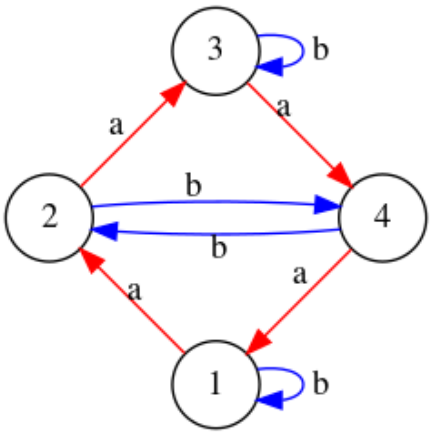
Por orden, ejecutamos el algoritmo y mostramos tanto la tabla de clases como el grafo de Schreier asociado.

[2]:
```
G = CosetTable(f)
G.CosetEnumeration()
```

[3]:
```
print(G.coset_table())
```

C	a	A	b	B
1	2	4	1	1
2	3	1	4	4
3	4	2	3	3
4	1	3	2	2

[4]:
```
G.schreier_graph(notes=False)
```



Se tiene que:

$$4 = [G : H] = \frac{|G|}{|H|} = \frac{|G|}{2}, \text{ por lo que } |G| = 8\,.$$

A continuación, construímos el grupo a partir de los generadores de Schreier y veremos que es isomorfo al grupo Diédrico D_4.

[5]:
```
generators = G.getGenerators()
S = Group(elems=generators)
print(S)
```

[5]:
```
Group with 8 elements: {(1, 2, 3, 4), (), (1, 2)(3, 4), (1, 4, 3,␣
↪2), (1, 4)(2, 3), (2, 4), (1, 3), (1, 3)(2, 4)}
```

[6]:
```
D4 = DihedralGroup(4)
D4.is_isomorphic(S)
```

[6]:
```
True
```

El objetivo ahora es comprobar que el grupo D_4 es producto semidirecto interno de dos de sus subgrupos. Véase el Ejemplo 9. En primer lugar, consideremos los subgrupos de D_4 generados por $R1$ y por $S0$:

[7]:
```
R = D4.generate(['R1'])
S = D4.generate(['S0'])
print(R, S)
```

[7]:
```
Group with 4 elements: {'R3', 'R2', 'R0', 'R1'}
Group with 2 elements: {'S0', 'R0'}
```

Ahora bien, estudiamos las acciones $\varphi\colon \langle S0\rangle \to \langle R1\rangle$. Por el Comentario 4.2, hay 2 posibles productos semidirectos (uno de ellos generado por la acción trivial).

[8]:
```
AutR = R.AutomorphismGroup()
Hom = S.AllHomomorphisms(AutD)
```

[9]:
```
hom0 = GroupHomomorphism(S, AutR, lambda x:Hom[0](x))
hom1 = GroupHomomorphism(S, AutR, lambda x:Hom[1](x))

SP0 = R.semidirect_product(S,hom0) #Hom0 es la acción trivial.
SP1 = R.semidirect_product(S,hom1) #Hom1 NO es trivial.
```

Bastará ver que el grupo Diédrico D_4 es producto semidirecto cuando la acción tomada no es la trivial, es decir:

$$D_4 \cong \langle R1\rangle \rtimes_{hom1} \langle S0\rangle\,.$$

[10]:
```
SP0.is_isomorphic(D4)
SP1.is_isomorphic(D4)
```

[11]:
```
False, True
```

Por último, consideramos dos grupos G y H definidos como sigue:

$$G = \langle a, b, c \mid a^6 = b^2 = c^2 = 1, abc \rangle \quad y \quad H = \{b\}.$$

La definición del grupo se encuentra en el fichero *Groups/3gens.txt*, y en principio no parece ser un grupo conocido.
Por orden: leemos el grupos haciendo uso del método *readGroup*, llamamos al método que ejecuta el *Algoritmo de Todd Coxeter*, y después vemos el número de clases que resultan en la tabla de clases de G/H o el número de vértices del grafo de Schreier (no los mostramos por espacio), obteniendo que $[G : H] = 6$.
El conjunto de generadores del grupo son:

$$g0 = (1,2,3,4,5,6),$$
$$g1 = (2,6)(3,5),$$
$$g2 = (1,6)(2,5)(3,4).$$

Definimos el grupo $G = \langle g0, g1, g2 \rangle$, es decir, aquel generado por los elementos $g0, g1$ y $g2$, obteniendo:

[1]:
```
print(G)
```

```
Group with 12 elements: {(1, 5)(2, 4), (1, 2, 3, 4, 5, 6), (1,␣
↪4)(2, 5)(3, 6), (1, 6)(2, 5)(3, 4), (1, 3)(4, 6), (1, 4)(2, 3)(5,␣
↪6), (), (2, 6)(3, 5), (1, 2)(3, 6)(4, 5), (1, 6, 5, 4, 3, 2), (1,␣
↪3, 5)(2, 4, 6), (1, 5, 3)(2, 6, 4)}
```

[2]:
```
group.is_abelian()
```

[2]: `False`

El grupo no es abeliano, luego debe ser isomorfo a uno de los siguientes grupos:

$$G \cong A_4 = \{a, b \mid a^3 = b^3 = (ab)^2 = 1\},$$
$$G \cong D_6 = \{a, b \mid a^6 = b^2 = 1, ab = a^{-1}b\},$$
$$G \cong Q_3 = \{a, b \mid a^6 = 1, a^3 = b^2, ab = a^{-1}b\}.$$

[3]:
```
A = AlternatingGroup(4)
D = DihedralGroup(6)
Q = QuaternionGroupGeneralised(3)

print(group.is_isomorphic(A))
print(group.is_isomorphic(D))
print(group.is_isomorphic(Q))
```

[3]: `False, True, False`

Como hemos visto, el grupo G se trata del grupo Diédrico D_6. Como consecuencia, vemos que la presentación de un grupo no es única.

5.3.4 *Otras presentaciones*

Para acabar, consideraremos grupos finitamente presentados que no se han estudiado en este proyecto. El objetivo es el de mostrar la potencia que tiene este método programado, llegando incluso a terminar en poco tiempo con ejemplos de grupos de orden muy alto.

1. Consideramos el grupo especial lineal $SL_n(F) := \{A \in M_n(F) \mid det(A) = 1\}$ y su centro $Z(SL_n(F))$. Definimos el grupo lineal especial proyectivo $PSL_n(F)$ como el cociente entre $SL_n(F)$ por $Z(SL_2(F))$.

   Consideramos un campo finito con 7 elementos, por ejemplo $\mathbb{Z}_7$, entonces $PSL_2(\mathbb{Z}_7)$ viene definido por:

   $$G = \langle a, b, c \mid a^7 = b^3 = c^2 = 1, ba = aab, (bc)^2, (ac)^2 \rangle.$$

   Este grupo se encuentra definido en el fichero *Groups/PSL2.txt* y se tomará el grupo trivial H para su ejecución. A pesar de ser un grupo de orden 168, termina en torno a 15 segundos con el método *HLT*.

2. Sea G un grupo generado por tres elementos a, b y c que satisfacen las siguientes relaciones:

   $$a^3 = b^2 = c^2 = 1,\ (ab)^4 = (ac)^2 = (bc)^3 = 1\,.$$

   y $H \leq G$ generado por a y b, es decir:

   $$G = \langle a, b, c \mid a^3 = b^2 = c^2 = 1,\ (ab)^4 = (ac)^2 = (bc)^3 = 1 \rangle \quad y \quad H = \langle a, b \rangle.$$

   Se encuentra disponible en *Group/Go.txt* y se trata de un grupo de orden 576. Su ejecución con el método programado tarda en torno a 1 minuto. De igual modo, es costoso generar todo el grupo de Permutaciones y realizar operaciones con los métodos de la librería.

3. El siguiente grupo es conocido como grupo de Mathieu, descubierto a finales del siglo XIX por el matemático francés Émile Mathieu junto a otros cuatro grupos de permutaciones. Se denota por M_{12} y viene dado por:

   $$M_{12} = \langle a, b, c \mid a^{11} = b^2 = c^2 = 1, (ab)^3 = (ac)^3 = (bc)^{10} = 1, a^2(bc)^2a = (bc)^2 \rangle.$$

   Se encuentra definido en el archivo *Groups/Big.txt*, y como subgrupo H, se ha tomado el trivial. Tras aplicar el *Algoritmo de Todd Coxeter*, obtenemos que el índice $[G : H] = |G| = 95040$. Requiere en torno a 600.000 clases laterales y su ejecución con el método *HLT* se demora en torno a 5 minutos. Por esta razón, no es pensable obtener los generadores de Schreier ni definir el grupo con estos ya que cualquier operación básica que se realice consumiría mucho tiempo.

Parte III

CONCLUSIONES Y TRABAJO FUTURO

6

CONCLUSIONES Y TRABAJO FUTURO

En primer lugar, se han introducido los principales conceptos y teoremas sobre la construcción de grupos libres, con los cuales podremos definir grupos dados por un conjunto de generadores y relatores, lo que se conoce por presentación de grupo.

Por otro lado, se ha presentado el producto semidirecto, una alternativa al producto directo de grupos en el que se obtiene un grupo a partir de dos más pequeños. Su construcción requiere de acciones de grupo por lo que se han introducido los conceptos más relevantes. Usando este producto semidirecto, se han descrito algunas técnicas de clasificación de grupos y se ha visto que es una herramienta potente y a veces más útil que el producto usual, pero no siempre es válida ya que no todo grupo puede expresarse como producto semidirecto (interno) de dos de sus subgrupos.

En relación con la parte informática, se ha realizado una extensión y optimización de la librería de Teoría de Grupos de José L. Bueso Montero y Pedro A. García Sánchez, una librería que se presenta como material didáctico y complementario a la Teoría de Grupos estudiada durante la carrera. Se han estudiado los principales problemas de la librería y se han aportado e implementado soluciones a éstos, como por ejemplo la implementación en clases de grupos que se definen de forma axiomatizada (asociatividad, identidad e inversos). Ante la imposibilidad de definir grupos dados por una presentación, se ha implementado el *Algoritmo de Todd Coxeter*, obteniendo así mucha información del grupo G, la más importante la representación por permutaciones del grupo G.

La versión implementada es conocida como el método *HLT* o *método basado de relatores*. Como hemos visto anteriormente, cumple con las necesidades de cualquier curso avanzado de Teoría de Grupos; sin embargo, si queremos profundizar y mejorar la eficiencia, se podrían implementar las siguientes estrategias, disponibles en [DBO06].

- Existe una variante del método *HLT* llamada *HLT+lookahead*, que puede ser útil cuando la tabla de clases es excesivamente grande y no disponemos de suficiente memoria. Supongamos que estamos trabajando con una clase x y el número de clases vivas excede el máximo valor de memoria permitido. En vez de abortar el proceso, el método opera sobre las clases definidas para buscar coincidencias y borrar aquellas clases que estén en la clase de equivalencia de otra que represente una clase menor. De este modo, liberamos memoria y permitimos al método seguir ejecutándose sobre la clase x.

- Por último, se describirá el método *Felsch*. En vez de forzar el escaneo de cada relator para cada clase como se realiza en ambos métodos anteriores, este método se centra en rellenar la tabla de clases de G/H del siguiente modo. Para los generadores del subgrupo H, se escanea y completa la clase 1. Para las relaciones que definen al grupo G, dada una clase x, se realizan definiciones para completar la fila de la clase x, y después de cada definición, se escanean todas las clases sin completar con nuevas definiciones. De esta forma, intentamos evitar las coincidencias, lo que se refleja en un menor uso de clases usadas.

Sigamos ahora con el estudio de nuevos algoritmos que pueden ser incorporados a la librería. Sea G un grupo finitamente presentado y $H \leq G$ un subgrupo suyo no trivial dado mediante generadores. Puede haber ocasiones en las que tras haber aplicado el *Algoritmo de Todd Coxeter* sobre G y H, nos encontremos que el índice $[G : H]$ es excesivamente grande y los generadores de H no nos aporten información relevante sobre el subgrupo. Por ello, podemos pensar en implementar algoritmos que puedan calcular una presentación del subgrupo H. Este tema se le atribuye a George Havas [Hav73] y Joachim Neubüser [Neu81], y gracias a su trabajo, existen hoy en día algoritmos que realizan esta función, como es el caso del *Algoritmo de Reidemeister-Schreier*. Este algoritmo requiere del *Algoritmo de Todd Coxeter* ya que usa los generadores de Schreier de G como datos de entrada. Véase [DBO06] para una descripción detallada.

Aquellas presentaciones de subgrupos obtenidas a partir del *Algoritmo de Reidemeister-Schreier* usan un número tan elevado de relatores que su manejo de vuelve dificultoso. Por ello, podemos aplicar transformaciones a los relaciones (transformaciones de Tietze) para obtener, en una secuencia finita de transformaciones, una presentación con un menor número de generadores y sin relatores redundantes.

Para terminar con nuestro estudio de algoritmos para grupos finitos, puede resultar de interés calcular una presentación de un grupo dado. Para grupos de orden pequeño carece de importancia, sin embargo, para grupos grandes es imprescindible, tanto para su manejo matemático como para su almacenamiento y/o representación computacional. Actualmente, existen algoritmos que hacen esto posible. El más importante se conoce como *Todd-Coxeter Schreier-Sims* [Mur03], un algoritmo que gira en torno a una presentación candidata a la que se le van añadiendo nuevos relatores y aplicando enumeración de clases para comprobar en cada paso si dicha presentación es isomorfa a la presentación del grupo original.

Los principales sistemas de desarrollo de software algebraico como GAP y Magma incluyen implementaciones de los algoritmos anteriores y de la gran mayoría de herramientas de Teoría de Grupos computacional. En relación con el *Algoritmo de Todd Coxeter*, GAP incorpora dos paquetes especiales, llamados ACE [GHHR01] ("Advanced Coset Enumerator") e ITC [VHJ00] ("Interactive Todd–Coxeter"). Este último tiene una interfaz gráfica y permite al usuario controlar el proceso paso a paso, experimentando con las diferentes estrategias explicadas anteriormente.

BIBLIOGRAFÍA

[Bat12] Michael Batty. *Essential Group Theory*. Bookboon, (2012). pp. 40-63. Disponible en Batty.

[BHLY15] P. D. Balsdon, A. Horawa, M. Lenain, and H. Yang. *Free Groups and Geometry*. Universidad de Michigan. (2015). pp. 1-7. Disponible en Horawa.

[BL18] M. Bullejos Lorenzo. *Álgebra II, apuntes del curso 18/19*. Universidad de Granada, (2018).

[BM13] F. Barrera Mora. *Introducción a la Teoría de Grupos*. Universidad Autónoma del Estado de Hidalgo (UAEH), (2013). pp. 40-86. Disponible en Barrera.

[Bue15] José L. Bueso. *Grado en Matemáticas: Álgebra II*. Universidad de Granada, (2015). pp. 35-86. Disponible en Bueso.

[CBFS14] John Cannon, Wieb Bosma, Claus Fieker, and Allan Steel. *Handbook of Magma functions*. Version 2.20. University of Sidney, (2014). Disponible en Magma.

[DBO06] F. Derek, Eick. Bettina, and Eamonn O'brien. *Handbook of computational group theory*. Discrete Mathematics and Its Applications, Chapman & Hall. CRC Press, (2006). pp. 149-217.

[DF03] David S. Dummit and Richard M. Foote. *Abstract Algebra*. John Wiley, (2003). pp. 152-215.

[GAP86] GAP. *Groups, Algorithms and Programming*. The GAP group. (1986). Disponible en www.gap-system.org/.

[GHHR01] Greg Gamble, Alexander Hulpke, George Havas, and Colin Ramsay. *GAP package ACE*. The GAP group. (2001). Disponible en ACE.

[GS13] Pedro A. García Sanchez. *Historia de las matemáticas: Teoría de Grupos*. Universidad de Granada, (2013).

[GSBM16] Pedro A. García Sánchez and José Luis Bueso Montero. *Librería de Teoría de Grupos*. Universidad de Granada, (2016). Disponible en Github.

[Har12] Naftali Harris. *Librería de Teoría de Grupos*. (2012). Disponible en Github.

[Hav73] George Havas. *A Reidemeister Schreier Program*. Proceedings of the Second International Conference on the Theory of Groups. Australian National University, Canberra, Springer-Verlag, Berlin, (1973). pp. 347-356.

[HR94] George Havas and Edmund F. Robertson. *Application of Computational Tools for Finitely Presented Groups*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Cambridge University Press, (1994). Disponible en arxiv.org.

[IC99] C. Iborra Castillo. *Álgebra y Geometría*. Universidad de Valencia, (1999). pp. 235-250. Disponible en Iborra.

[JM17] Pascual Jara Martínez. *Teoría de Grupos: Estructura de grupos finitos*. Universidad de Granada, (2017). pp. 155-182. Disponible en www.ugr.es.

[Joh90] D.L. Johnson. *Presentations of Groups*. London Mathematical Society Student Texts. Cambridge University Press, (1990). pp. 100-135.

[Jud17] Thomas W. Judson. *Abstract Algebra: Theory and Applications*. Orthogonal Publishing L3c, (2017). Cap. 14,15. Disponible en Abstract.

[KB70] D. Knuth and P. Bendix. *Simple word problems in universal algebras*. Oxford University. Pergamon Press, (1970).

[MFE03] M. Ángeles Moreno Frías and E. Pardo Espino. *Teoría de grupos*. Textos Universidad de Cádiz Series. Servicio de Publicaciones de la Universidad de Cádiz, (2003).

[Mil16] Kyle Miller. *The Todd-Coxeter algorithm*. Universidad de Berkeley, (2016). Disponible en Kmill.

[Mil20] James S. Milne. *Group Theory*. (2020). pp. 31-42, 57-74. Disponible en www.jmilne.org/math/.

[Mur03] Scott H. Murray. *The Schreier-Sims algorithm*. Department of mathematics. Australian National University, (2003). pp. 25-40. Disponible en Scott.

[Neu81] J. Neubüser. *An elementary introduction to coset-table in computational group*. London Mathematical Society. Cambridge University Press, (1981). pp. 1-45.

[Pal18] E. Miranda Palacios. *Álgebra*. Universidad de Granada, (2018).

[Sim05] Stephen G. Simpson. *A Slick Proof of the Unsolvability of the Word Problem for Finitely Presented Groups*. (2005). Disponible en Word Problem.

[TC36] J. A. Todd and H. S. M. Coxeter. *A practical method for enumerating cosets of a finite abstract group*. Universidad de Manchester & Cambridge, (1936). Disponible en Todd Coxeter.

[VHJ00] Felsch Volkmar, Ludger Hippe, and Neubüser Joachim. *GAP package ITC*. The GAP group. (2000). Disponible en ITC.