Capítulo 1

Enumeración de cosets.

Dado un grupo G finitamente presentado muchas de las preguntas más básicas que nos podemos hacer sobre el grupo tales como ″¿es finito?.º más aún ″¿es el grupo trivial?resultan ser no decidibles. El problema de la palabra que es el de ″decidir si dos palabras en los generadores del grupo son idénticas″también resultó ser no decidible.

Dicho esto muchos de estos problemas resultan ser semidecidibles. Esto es que si resulta que este grupo es trivial o finito lo podemos probar si bien no tengamos un algoritmo que funcione para este caso.

El procedimiento que vamos a enfocarnos toma un grupo G finitamente presentado y un subgrupo H tal que sus generadores están especificados como palabras en los generadores de G. Este procedimiento intenta verificar que $(G : H)$ es finito. No podemos decir que es un algoritmo porque no siempre va a terminar. Si $(G : H)$ es finito entonces va a terminar. En cambio si es infinito en principio el procedimiento correría para siempre (en la práctica hasta que se termine la memoria de la computadora). Si el procedimiento termina habremos obtenido que el índice es finito y obtendremos cuánto es este índice y como es la tabla de los cosets. Esta tabla de cosets resulta ser equivalente a la representación de G como permutaciones de los cosets de H a derecha visto con la multiplicación a derecha.

§1.1. Ejemplo a mano.

Veamos primero un ejemplo de como funciona el algoritmo en un caso chiquito. Para facilitarnos la lectura cada vez que tenemos un generador a vamos a escribir $\overline{a}$ cuando estamos hablando de a^{-1}.

Consideremos el grupo finito dado por la presentación $G = \langle x, y \mid x^2, y^3, xyx^{-1}y^{-1}\rangle$ que es isomorfo al grupo $C_2 \times C_3$. Consideremos que $H = \langle x\rangle$. El algoritmo entonces lo que nos debería devolver es $|G : H| = 3$.

Empezamos denotando con **1** al coset trivial correspondiente a H en G/H. Como $x \in H$ entonces sabemos que $Hx = H$ denotaremos por $1^x = 1$ a la multiplicación de 1 por x. Por otro lado denotemos con **2** al coset 1^y y denotaremos por **3** al coset $1^{\overline{y}}$.

¿Qué podemos hacer? Sabemos que todo coset α es tal que $\alpha^r = \alpha$ para todo $r \in R$. Vamos a

introducir un proceso que se llama *escanear las relaciones*.

Por ejemplo tomemos $r = y^3$ y escaneamos a 1. Tenemos que $1^y = 2$ por como definimos al coset 2. Por otro lado tenemos que $1^{y^2} = 2^y$ todavía no está definido. Como $r^{-1} = 1$ visto como elemento en G si escanemos ahora viendo esta relación tenemos que $1^{\overline{y}} = 3$ porque así lo definimos y por otro lado que $3^{\overline{y}}$ no está definido. De esto podemos *deducir* que $2^y = 3$. Una manera gráfica de ver este proceso de deducción es la siguiente:

Agregar grafiquito como un grafo que nos da la deducción.

Notemos que de esta deducción también obtenemos que $3^{\overline{y}} = 2$.

Finalmente si escaneamos las otras relaciones para los cosets 1 y **2** obtenemos que $3^x = 3$ y que $2^x = 2$.

De esta manera podemos juntar todo lo obtenido en la siguiente tabla:

Como todos los cosets escanean correctamente por medio de las relaciones del grupo y todas las entradas de la tabla de cosets están bien definidas el procedimiento termina y obtenemos que el indice del subgrupo resulta ser equivalente a la cantidad de cosets que nos terminaron quedando, en este caso exactamente 3.

§1.2. Tablas de cosets.

Ahora veamos de definir la tabla en el caso más en general.

Partamos de un grupo finitamente presentado G con presentación $\langle X|R\rangle$ y un subgrupo H que es finitamente generado por $\langle Y\rangle$. Haremos la distinción nombrando $A = X \cup X^{-1}$ a los generadores como monoide. A los conjuntos R e Y los vamos a considerar como subconjuntos de palabras en A^*.

Definimos la tabla de cosets $\mathcal{C}$ como una tupla con los siguientes 4 elementos.

1. n un número natural;
2. $\tau : [1 \dots n] \to A^*$ una función;
3. $p : [1 \dots n] \to [1 \dots n]$ una función tal que $p(\alpha) \leq \alpha$ para todo $\alpha \in [1 \dots n]$.
4. $\chi : [1 \dots n] \times A^* \to [1 \dots n]$ una función parcial –no necesariamente definida para todas las tuplas (α, w) con $\alpha \in [1 \dots n]$ y $w \in A^*$.

Todos estos atributos los iremos cambiando a medida que corremos el algoritmo.

La función parcial χ es lo que nos va a dar la multiplicación por los generadores A en los cosets. Generalmente escribiremos para referirnos a $\chi(\alpha, x) = \beta$ como $\alpha^x = \beta$ así tal como hicimos en el ejemplo anterior.

Definimos los cosets vivos $\Omega = \{\alpha \in [1 \dots n] \mid p(\alpha) = \alpha\}$. El rol de p como función es diferenciarnos qué números de $[1 \dots n]$ efectivamente se corresponden a cosets que, hasta el momento, sepamos que son cosets distintos. Esto es porque muchas veces sucede en el algoritmo que nombraremos con $\alpha < \beta$ a dos elementos que resultaron ser efectivamente el mismo coset.

La tabla de cosets vamos a querer que mantenga ciertas propiedades a medida que corramos el algoritmo, esto vendría a ser algo así como el invariante de representación. A todo momento queremos que valgan las siguientes cuatro propiedades.

P1. $1 \in \Omega$ y $1^{\tau(\alpha)} = \alpha$.

P2. $\alpha^x = \beta \iff \beta^{\overline{x}} = \alpha$.

P3. Si $\alpha^x = \beta$ entonces tenemos que $H\tau(\alpha)x = H\tau(\beta)$

P4. Para todo $\alpha \in \Omega$ tenemos que $1^{\tau(\alpha)}$ está definido y $1^{\tau(\alpha)} = \alpha$.

§1.3. Algoritmos para definir y recorrer cosets.

Veamos ahora como son los algoritmos que tenemos para ir llenando la tabla de cosets.
Primero veamos el primer algoritmo que nos sirve para definir un coset nuevo.

Agregar el seudocódigo de este algoritmo.

Notemos que si suponemos que las propiedades **P1-P4** valen entonces si corremos el algoritmo de definir siguen valiendo.

Ahora veamos el procedimiento de escanear un coset bajo una palabra arbitraria w.

Cuando estamos escaneando un coset por una palabra podemos llegar a una *deducción* tal como hicimos en los casos anteriores. Veamos que estas deducciones no nos rompen las propiedades que queremos que mantengan las tablas de cosets.

PROPOSICIÓN 1.3.1. *Si deducimos que $\alpha^x = \beta$ y que $\beta^{\overline{x}} = \alpha$ a partir de escanear algún coset α para alguna relación $w \in R$ o bien $w \in Y$ luego todas las propiedades **P1-P4** siguen siendo ciertas después de esta deducción.*

Demostración. . ■

§1.4. Correctitud del algoritmo (caso sin coincidencias).

Probemos la correctitud de este procedimiento en este caso que no tenemos coincidencias. Vale el siguiente teorema.

TEOREMA 1.4.1. *Asumamos que:*

1. *Las propiedades **P1-P4** son válidas;*
2. *La tabla de cosets está completa (dicho de otra forma);*
3. *El **1** escanea correctamente para todo $w \in Y$;*
4. *Todos los $\alpha \in [1 \dots n]$ escanean correctamente para toda palabra $r \in R$.*

Entonces $|G : H| = |\Omega|$. Más aún para cada $x \in A$ obtenemos $\phi_x(\alpha) : \Omega \to \Omega$ tal que $\phi_x(\alpha) = \alpha^x$ es una permutación de Ω y podemos extenderlo a ϕ un morfismo $G \to S(\Omega)$ que es equivalente a la acción de G sobre los cosets G/H.

Demostración. . ■

§1.5. G-congruencias.

En esta sección vamos a considerar un grupo $G = \langle X, R \rangle$ finitamente generado como en el resto del algoritmo aunque vamos a tomar las palabras en el monoide libre de los generadores $A = X \cup X^{-1}$. Queremos escribir en qué contexto más general el grupo G se relaciona con los cosets en un momento dado del algoritmo. Sabemos que al finalizar el algoritmo nos queda la acción sobre los cosets G/H si nos restringimos a Ω pero ¿qué sucede en el medio? Para esto es necesario hablar de G-congruencias.

DEFINICIÓN 1.5.1. Un conjunto X que tiene una acción de G se llamará un G-conjunto.

DEFINICIÓN 1.5.2. Una G-congruencia es una relación de equivalencia sobre un G conjunto X que aparte satisface la siguiente propiedad. Si $x \sim y$ luego para todo $g \in G$ vale que $xg \sim yg$.

Así como tenemos relaciones de equivalencias generadas a partir de algún subconjunto $S \subseteq X \times X$ podemos considerar la menor G congruencia generada a partir de S. Tomamos la intersección de todas las G-congruencias que contengan a S. Sabemos que esta intersección es no trivial porque siempre podemos tomar la G congruencia donde todos los elementos están relacionados con todos. Otra manera de definir a esta menor congruencia es por medio de la propiedad universal. Sea $\sim_S$ esta relación y sea $\sim$ otra relación tal que $\alpha \sim \beta \implies \alpha \sim_S \beta$. En este caso tenemos que

Interpretación con la propiedad universal.

DEFINICIÓN 1.5.3. Dada una G-congruencia $(X, \sim)$ y un par $(\alpha, \beta) \in X \times X$ podemos considerar a la G congruencia generada por este par a partir de $\sim$. [1]

Dado un conjunto X con una G-congruencia es evidente que G actúa en X porque X es un G conjunto. Podemos ver que esta misma acción baja al cociente.

LEMA 1.5.4. *La acción de G sobre X se restringe a una acción de G sobre $X/\sim$.*

Demostración. Queremos ver que la siguiente aplicación es una acción,

$$G \times X/\sim \to X/\sim$$
$$[x]g = [xg],\ \forall g \in G$$

[1] Estamos tomando la menor G-congruencia que contenga a $S \cup (\alpha, \beta)$ donde S son todos los pares que definen la relación $\sim$.

Como $\sim$ es una G congruencia tenemos que $[x]g = [y]g$ si y solo si $[x] = [y]$ entonces está bien definida la aplicación. La buena definición de la acción de G sobre X nos dice que está bien definida la acción sobre el cociente $X/\sim$. ■

Si bien la definición anterior la dimos para una acción de G en nuestro caso en particular nos va a bastar que G tenga una acción parcial. Esto es que exista una función parcial.

$$\rho : X \times G \to X$$

donde $\rho(x, g)$ estará definida para algunos pares $(x, g) \in X \times G$ no necesariamente para todos. Nuestra definición de una G congruencia la podemos modificar entonces para que $x \sim y \implies xg \sim yg$ siempre y cuando xg y yg estén ambas definidas.

Sea $([1, \dots, n], \sim)$ la G congruencia generada por p. Esto es que para $\kappa, \lambda \in [1, \dots, n]$

$$\kappa \sim \lambda \iff \text{REP}(\kappa) = \text{REP}(\lambda).$$

Por como definimos a $\Omega = \{\alpha \in [1, \dots, n] \ : \ p(\alpha) = \alpha\}$ tenemos que $\Omega \simeq [1, \dots, n]/\sim$.

§1.6. Algoritmo de coincidencias.

Cuando llegamos a que dos cosets $\alpha, \beta \in [1, \dots, n]$ que inicialmente consideramos diferentes son tales que resultan ser iguales corremos el algoritmo de COINCIDENCIA(α, β).

El siguiente seudocódigo nos da una idea de como funciona la rutina.

§1.6.1. Demostración de la correctitud del algoritmo de coincidencias.

El algoritmo de coincidencias usa MERGE y REP, de manera que modifica el valor de la función p. Cuando terminamos de correr el algoritmo tenemos una función p' definida sobre los cosets.

LEMA 1.6.1. *La función $p' : [1, \dots, n] \to [1, \dots, n]$ cumple que $p'(\alpha) \leq \alpha$ para todo $\alpha \in [1, \dots, n]$.*

Demostración. En los únicos momentos que reducimos el valor de p es cuando llamamos a REP o bien a MERGE en las líneas (...)

El algoritmo MERGE solo reduce el valor de p porque justamente toma el coset que tenga p más bajo y redefine al otro para que tenga el mismo valor.

Por otro lado el algoritmo REP podemos ver usando inducción directamente que reduce el valor de p porque justamente p es decreciente y al aplicarla sucesivamente nos queda que $p^i(\alpha) \leq \alpha$ para todo $i \in \mathbb{N}$.

En definitiva p' nos queda una función decreciente y definida sobre el mismo intervalo de enteros tal como queríamos ver. ■

Llamemos $\sim_F$ la relación que nos queda a partir de p' al terminar de hacer COINCIDENCIA. Esto es que $\kappa \sim_F \lambda$ sii al terminar el algoritmo $\text{REP}(\kappa) = \text{REP}(\lambda)$ donde en este caso el representante lo estamos mirando respecto a esta nueva relación de equivalencia. Podemos probar que la propiedad que vale anteriormente para la relación $\sim$ sigue valiendo para nuestra nueva relación $\sim_F$.

LEMA 1.6.2. *Si existen $\gamma, \delta \in [1, \dots, n], x \in A$ tales que $\gamma^x = \delta$ antes de comenzar el algoritmo entonces al terminarlo vale que* $\text{REP}(\gamma)^x = \text{REP}(\delta)$.

Demostración. Esta propiedad sabemos que vale al iniciar el algoritmo para la relación $\sim$ justamente por lo probado en el lema 1.5.4. En el caso que $p(\gamma) = p'(\gamma)$ y que $p(\delta) = p'(\delta)$ como vale para $\sim$ va a valer para $\sim_F$. Esto es porque el representante sigue siendo el mismo en ese caso. Por lo tanto solo nos interesa los casos que p' es distinta de p. Siguiendo el algoritmo tenemos tres casos posibles en los que modificamos el valor de p'. Así como en el algoritmo tomamos $\mu = \text{REP}(\gamma)$ y $\nu = \text{REP}(\delta)$.

(I). Si $\mu^x = \lambda$ entonces hacemos $\text{MERGE}(\lambda, \nu)$. Esto hace que ahora $\text{REP}(\lambda) = \text{REP}(\nu)$ por lo tanto $\text{REP}(\gamma)^x = \mu^x = \lambda$ y por otro lado vimos que $\lambda \sim \nu$ por lo que $\text{REP}(\delta) = \text{REP}(\lambda)$ y así terminamos de probar lo que queríamos ver.

(II). Similar al caso anterior tomando $\nu^{x^{-1}} = \kappa$.

(III). Este último caso es el que no esté definido μ^x ni $\nu^{x^{-1}}$ y lo definimos directamente de manera que $\mu^x = \nu$ y en particular $\text{REP}(\gamma)^x = \text{REP}(\delta)$.

■

Finalmente podemos probar lo que queríamos ver. Al terminar el algoritmo nos sigue quedando una G congruencia. Más aún esta G congruencia la conocemos y es justamente la generada por el par (α, β).

TEOREMA 1.6.3. *Al terminar del correr el algoritmo* $\text{COINCIDENCIA}(\alpha, \beta)$ *obtenemos la siguiente igualdad para todo $\kappa, \lambda \in [1, \dots, n]$,*

$$\kappa \sim_F \lambda \iff \kappa \equiv \lambda.$$

OBSERVACIÓN 1.6.4. En particular esto nos dice que $\sim_F$ que en principio solo es una relación de equivalencia resulta ser una G congruencia.

Demostración. Primero veamos que $\lambda \sim_F \kappa \implies \kappa \equiv \lambda$. Ni bien comienza el algoritmo sabemos que vale esta implicación porque justamente $\equiv$ es la G-congruencia generada a partir de $\sim$ y del par (α, β). Los casos que llamamos a MERGE son los casos que podría modificarse la relación $\sim$. El primer MERGE justamente agrega al par (α, β) a la relación $\sim$ y sabemos que este par está por definición en $\equiv$. Para los otros llamados de MERGE tenemos que $\gamma^x = \delta$ tal que podemos suponer inductivamente que $\gamma \sim \mu = \text{REP}(\gamma)$ y que $\delta \sim \nu = \text{REP}(\delta)$ de manera que ahora agregamos $\mu^x \sim \nu$. Como $\equiv$ es una G-congruencia vale que $\mu^x \equiv \gamma^x$ porque suponemos inductivamente que $\mu \equiv \gamma$. Si estamos en el caso que está definido $\mu^x = \lambda$ obtenemos que $\mu^x \equiv \nu$. Análogamente el caso que $\nu^{x^{-1}}$ está definido. En ambos casos vimos que si $\kappa \sim \lambda \implies \kappa \equiv \lambda$. Como esto vale en todo paso del algoritmo vale más aún cuando termina así que terminamos de ver esta implicación.

Veamos ahora la otra implicación. En vez de probarla directamente usemos que $\equiv$ es la menor G-congruencia que contiene a $\sim$ y al par (α, β). Si vemos que $\sim_F$ es una G-congruencia entonces terminaríamos de ver que $\kappa \sim_F \lambda \iff \kappa \equiv \lambda$ tal como queríamos ver.

Para eso partamos de $\kappa \sim_F \lambda$ y veamos que si $x \in A$ es tal que κ^x y λ^x estén ambas definidas entonces $\kappa^x \sim_F \lambda^x$. Haciendo inducción en la longitud de la palabra podemos probarlo para $w \in A^*$ genérica por lo tanto alcanza con verlo para este caso de una sola letra.

Por el lema anterior 1.6.2 tenemos que al ser $\text{REP}(\kappa) = \text{REP}(\lambda)$ luego $\text{REP}(\kappa)^x = \text{REP}(\kappa^x)$ y similarmente $\text{REP}(\lambda)^x = \text{REP}(\lambda^x)$. De esta manera obtenemos que $\text{REP}(\kappa^x) = \text{REP}(\lambda^x)$ por lo tanto $\kappa^x \sim_F \lambda^x$. ■

PROPOSICIÓN 1.6.5. *Al terminar de correr* COINCIDENCIA(α, β) *todas las propiedades de la tabla de cosets* **??** *siguen valiendo.*

Demostración. Antes de correr el algoritmo todas estas propiedades valen. Cuando llamamos a COINCIDENCIA(α, β) estamos presuponiendo que $H\tau(\alpha) = H\tau(\beta)$ vistos como elementos en el conjunto de cosets a derecha G/H. Por un argumento inductivo similar al que hicimos anteriormente podemos ver que cada vez que hacemos MERGE(κ, λ) obtenemos que $H\tau(\kappa) = H\tau(\lambda)$.

P1. No modificamos en ningún momento $\tau(1)$ ni cambiamos $p(1)$ por lo tanto sigue valiendo al finalizar el algoritmo.

P2. El único caso que definimos una nueva entrada en la tabla de cosets sucede cuando tomamos $\mu^x = \nu, \nu^{x^{-1}} = \mu$ y esto hace que siga valiendo la propiedad. El caso que eliminamos a $\delta^{x^{-1}}$ de la tabla justamente es tal que δ deja de estar en los cosets vivos por lo tanto no afecta a esta propiedad.

P3. Vale por lo visto anteriormente porque al agregar $\mu^x = \nu, \nu^{x^{-1}} = \mu$ es el único momento que agregamos entradas a cosets vivos y por el argumento inductivo vemos que $H\tau(\mu^x) = H\tau(\nu)$.

P4. Sigue valiendo porque vale al comenzar para todos los cosets vivos y en particular vale para los que sobreviven.

■

§1.7. El algoritmo termina si el índice es finito.

El algoritmo de enumeración de cosets termina si el subgrupo que tomamos H es tal que $[G : H] < \infty$. Para probar esto vamos a considerar otra propiedad extra que queremos que cumpla nuestra tabla de cosets. A diferencia de las otras propiedades que podíamos interpretar como un invariante de representación del algoritmo esta último propiedad requiere de $\overline{\Omega}$ que va a ser el conjunto de los cosets que no son eliminados por ningún llamado de COINCIDENCIA. Notemos que no hay razón alguna por la que este conjunto sea finito a diferencia de Ω que por definición es siempre un subconjunto de $[1, \ldots, n]$ para $n \in \mathbb{N}$.

Propiedad 5. El conjunto $\overline{\Omega}$ es tal que

(I) Para cada $\alpha \in \overline{\Omega}$ y para cada $x \in A$ en algún momento α^x va a estar definido;

(II) Para cada $w \in Y$ en algún momento llamamos a SCAN(w);

(III) Para cada $\alpha \in \overline{\Omega}$ y para cada $w \in R$ en algún momento llamamos a SCAN(α, w).

Si tenemos que vale esta propiedad extra y que a todo tiempo valen las propiedades anteriores **??** entonces si el índice es finito el algoritmo termina.

TEOREMA 1.7.1. *Si las propiedades 1 a 5 se mantienen durante el algoritmo de enumeración de cosets de G con respecto al subgrupo H y $[G : H] < \infty$ entonces el algoritmo eventualmente termina.*

Demostración. Primero probemos que $\overline{\Omega}$ debe ser infinito sino el algoritmo termina y no hay nada que probar. Por la propiedad 5 sabemos que para todo $\alpha \in \overline{\Omega}$ es tal que en algún momento se define α^x supongamos $\alpha^x = \beta$. Por la definición de $\overline{\Omega}$ nunca eliminamos a α de Ω por lo tanto usando el resultado anterior **??** sabemos que $p[\alpha] = \alpha$ desde que lo definimos. Dado que α^x a lo sumo puede decrecer y al haber una cantidad finita de posibilidades eventualmente se debe estancar en cierto valor. Usando que bajo nuestra suposición $\overline{\Omega}$ es finito podemos tomarnos el máximo de todos los tiempos tales que α^x se estanca para todo $\alpha \in \overline{\Omega}$ y $x \in A$ obteniendo que en ese momento $\overline{\Omega} = \Omega$ con la tabla completa.

Nos queda ver el caso que $\overline{\Omega}$ es infinito. Consideremos la tabla de cosets infinita donde los cosets vivos son justamente $\overline{\Omega}$ y donde $\alpha^x = \beta$ para el valor en el que se estanque α^x que sabemos que está definido en cierto momento. Como vale la propiedad 5 estamos en las condiciones de usar el teorema **??** de manera que la acción de G por multiplicación a derecha sobre los cosets a derecha tiene orden infinito pero esto contradice que $[G : H] < \infty$. ■

§1.8. El método de las relaciones.

El siguiente método es uno de los más sencillos y directos de describir de los posibles métodos para enumerar cosets. La idea de este método es definir cosets cada vez que hacemos SCANy nos topamos con un posible coset nuevo sin definir con el propósito de terminar todos los scans. El siguiente algoritmo nos va a servir para escanear e ir completando la tabla a medida que la recorremos.

Primero implementamos un algoritmo sencillo que dado un coset y una palabra la recorre hasta donde pueda. Si α es el coset y la palabra es w (no necesariamente un generador de H ni una relación) entonces este algoritmo nos devuelve un par $(\beta, w_1) \in [1, \ldots, n] \times A^*$. La definimos de manera que $w = vw_1$ donde v es el mayor prefijo para el cual está definido α^v. Tomamos a $\beta = \alpha^v$ que es el coset al que llegamos leyendo todo lo que podemos leer de la palabra w.

El algoritmo opera de la siguiente manera. Como entrada tenemos un coset $\alpha \in [1, \ldots, n]$ y una palabra $w \in R$. Sabemos que bajo estas hipótesis $\alpha^w = \alpha$.

§1.9. Enumeración de cosets.

§1.10. Ejemplo de $C_2 * C_3$

Veamos de hacer el ejemplo sobre un grupo infinito.